REPORT DOCUMENTATION PAGE	*Form Approved OMB No. 0704-0188*

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE 26.Oct.98	3. REPORT TYPE AND DATES COVERED THESIS

4. TITLE AND SUBTITLE FUTURE LOCATION PREDICTION USING HIDDEN MARKOV MODELING	5. FUNDING NUMBERS
6. AUTHOR(S) 2D LT SCHNOOR MATTHEW A	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) UNIVERSITY OF COLORADO AT COLORADO SPRINGS	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) THE DEPARTMENT OF THE AIR FORCE AFIT/CIA, BLDG 125 2950 P STREET WPAFB OH 45433	10. SPONSORING/MONITORING AGENCY REPORT NUMBER 98-106
11. SUPPLEMENTARY NOTES	
12a. DISTRIBUTION AVAILABILITY STATEMENT Unlimited distribution In Accordance With AFI 35-205/AFIT Sup 1	12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

19981119 039

DTIC QUALITY INSPECTED 4

14. SUBJECT TERMS			15. NUMBER OF PAGES
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT

FUTURE LOCATION PREDICTION USING HIDDEN MARKOV MODELING

by

Matthew Allen Schnoor

B.S., United States Air Force Academy, 1997

A thesis submitted to the

University of Colorado in partial fulfillment

of the requirements for the degree of

Master of Engineering

College of Engineering and Applied Sciences

1998

Schnoor, Matthew Allen (Master of Engineering, Space Operations)

Future Location Prediction using Hidden Markov Modeling

Thesis directed by Dr. Charles E. Fosha

ABSTRACT

Mobility of mobile SCUD launchers in Desert Storm produced a need for improved methods of tracking and location. Without ways to track SCUD launchers, great amounts of time and resources will continue to be wasted on search and destroy in future conflicts.

Hidden Markov modeling provides a novel approach to predicting future movements of mobile SCUD launchers and other types of vehicles. The power of the hidden Markov model lies in the fact that it is a doubly stochastic process.

With this process we can not only model the output of the system but also the model underlying Markov states that the system will visit. Knowing the underlying state transitions can supply an added benefit in predicting future movements.

References:

[1] Atkinson on Scuds, http://ww2.pbs.org/wgbh/pages/frontline/gilf/weapons/ascud.html

[2] RadarSat Information, coorda@radar.space.gc.cs, Canadian Space Agency, 19 January 1996

[3] John T. Parish, personal interview, Sept 1997.

[4] Makhoul, John, and Richard Schwartz. "What is a hidden Markov model?" *IEEE Spectrum* Dec.1997:44-45.

[5] Fielding, Kenneth H., and Dennis W. Ruck. "Spatio-Temporal Pattern Recognition Using Hidden Markov Models." *IEEE Transactions on Aerospace and Electronic Systems* 31(1995):1292-1300.

[6] Baldi, Pierre, Yves Chavin, Tim Hunkapiller, and Marcella A. McClure. "Hidden Markov models of biological primary sequence information." *Proceedings of the National Academy of Sciences USA* 91(1994):1059.

[7] Hughey, Richard, and Anders Krogh. "Hidden Markov models for sequence analysis: extension and analysis of basic method." http://www.cse.ucsc.edu/research/c..mat/papers/hughkrogh96/cabious.html, 1996.

[8] Makhoul, John, Salim Roucos, and Herbert Gish. "Vector Quantization in Speech Coding." *Proceedings of the IEEE* 73(1985):1551.

[9] Morrison, Foster. "Model Validation." *The Art of Modeling Dynamic Systems*. New York: John Wiley and Sons, 1991. 347-357.

[10] Dr. Donald Caughlin, Personal Interview, March 1998.

[11] Hsu, Hwei P. "Random Signals and Noise." *Schwaum's outlines: Analog and Digital Communications*. New York: McGraw-Hill, 1993. 184-191.

[12] Hillier, Fredrick S., and Gerald J. Lieberman. "Markov Chains." *Introduction to Operations Research*. 6th ed. New York: McGraw-Hill, 1995. 628-660.

[13] Scheaffer, Richard L. *Introduction to Probability and its Applications*. Belmont: Duxbury Press, 1995. 6-68, 120-128, 225.

[14] Rabiner, Lawerence R. "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition." *Proceedings of the IEEE* (77) 1989:257-285.

[15] Dunning, Ted. "What makes a Hidden Markov Model hidden?" http://nora.hd.uib.no/carpora/1995-2/0047.html

[16] Levinson, S.E., L.R. Rabiner, and M.M. Sondhi. "An Introduction to the Application of the Theory of Probabilistic Functions of a Markov Process to Automatic Speech Recognition." *Proceedings of the IEEE* (62)1983:1035-1069.

[17] Forney, David G. Jr. "The Viterbi Algorithm." *Proceedings of the IEEE* (61)1972:268-273.

[18] Kiemele, Mark J., and Stephen R. Schmidt. *Basic Statistics: Tools for Continous Improvement*. 3rd ed. Colorado Springs, CO: Air Academy Press, 1993.3-28.

[19] Press, William H., Brian P. Flannery, Saul A Teukolosky, and William T. Vetterling. *Numerical Receipes in C*. Cambridge: University Press, 1986.

[20] Bevington, Philip R., and D. Keith Robinson. *Data Reduction and Error Analysis for the Physical Sciences*. 2nd ed. New York:McGraw-Hill, 1992. 96-110.

[21] Juang, B.H., and R.L. Rabiner. "A Probabilistic Distance Measure for Hidden Markov Models." *AT&T Technical Journal* (64)1985:391-402.

[22] User's Guide: MathCad 5.0 for Windows, 1994, pg 379.

This thesis for the Master of Engineering degree by

Matthew Allen Schnoor

has been approved for the

Department of

Aerospace Engineering

by

[illegible]

Charles E. Fosha, Jr.

[illegible]

Donald J. Caughlin, Jr.

[illegible]

Robert A.Rappold

Date 5/11/98

Schnoor, Matthew Allen (Master of Engineering, Space
Operations)
Future Location Prediction using Hidden Markov Modeling
Thesis directed by Dr. Charles E. Fosha

ABSTRACT

Mobility of mobile SCUD launchers in Desert Storm produced a need for improved methods of tracking and location. Without ways to track SCUD launchers, great amounts of time and resources will continue to be wasted on search and destroy in future conflicts.

Hidden Markov modeling provides a novel approach to predicting future movements of mobile SCUD launchers and other types of vehicles. The power of the hidden Markov model lies in the fact that it is a doubly stochastic process.

With this process we can not only model the output of the system but also the model underlying Markov states that the system will visit. Knowing the underlying state transitions can supply an added benefit in predicting future movements.

Dedication:

For my wife, Meg, my mom, and my brother, Mark.

You guys are the best. Thank you for everything.

CONTENTS

TABLES

FIGURES

CHAPTER I

INTRODUCTION

During the Gulf War, Iraq was thought to possess some 30 fixed SCUD sites. The problem with these sites was that Saddam never once launched a SCUD out of one of those. He preferred using mobile platforms consisting of a truck that could set up, launch a SCUD, and escape in a matter of minutes [1]. Even though they were militarily insignificant, political pressure forced many forces to be almost solely utilized for search-and-destroy missions against these targets. This presented a significant problem because western Iraq is about the size of Vermont, New Hampshire, and Massachusetts put together [1].

Needless to say, a better job could have been done in force utilization if the targets could have been found more efficiently. The problem then becomes: how can past movements of mobile SCUD's be correctly modeled to predict future movements and narrow search areas in future conflicts? This will decrease the time and amount of resources needed to destroy SCUD's. Fortunately, that is possible in the future if space assets are used wisely.

These assets can provide the essential initial inputs of environment data and past observations needed to predict future movements. Satellites like Canada's Radarsat can provide excellent moisture gradients using synthetic aperture radar (SAR) [2],[3]. Moisture content is important because mobile Scud's can't operate in too damp of soil. Other space-based assets can also provide remote sensing capabilities for topographical vector fields of terrain. Aggregating this information will provide a basis with which to work on predicting future movements of mobile platforms like SCUD launchers.

Predicting future movements of mobile launchers can hopefully be described by the use of hidden Markov models (HMM). The hidden Markov model has found most use in the area of speech recognition for computers but has recently found use in several different areas. Everything from target classification to computational biology now uses HMM [4],[5],[6],[7],[8]. This study will attempt to use a novel approach to HMM. Based on artificial observational data generated by a computer[1], an algorithm will be formulated to determine the crew's mindframe at any time iteration, the correct model order, and make predictions for possible future movements. The objective of the mobile SCUD is unknown, so the prediction of future movement has to come from past observed data. A full definition of the hidden Markov model can be seen as part of chapter II.

This approach is novel for two reasons. The first reason has to do with the time of analysis. This approach will determine useful information based on very few observations. There is no prior knowledge that the formulation will generate any useful information based on as

[1] No actual observations of bearing were available to use as data for this study.

few as three observations. Also, because of the requirement for timely prediction, it is important that the initial parameters of the model be accurate based on apriori knowledge. This will hopefully reduce the observations that are required to get information about the object.

The second novelty deals with how the model order is chosen. Past observations will be split into two groups. The data designated "distant" past will be used to parameterize the model. The model will then try to forecast the data designated "recent" past. The best model order will forecast the recent past with the highest probability [9]. Verification or order is knowing that a particular model predicts the recent past with a higher probability than all other models. In other words, the order that predicts the recent past best will be considered the proper order. No truth model is designated outright in this paper to generate data. Data will be generated from an ideal model through the computer.

It is important to note that available data will be restricted. With such small amounts of data, a truth model can mislead an observer over a small amount of time due to its probabilistic nature. Verification of model order under these circumstances would be difficult. Therefore, a truth model is not used for verification. The ideal model is one that would predict the recent past with perfect accuracy. The order verification will come in how close the given model is to an ideal model, not the truth model.

Although this paper emphasizes mobile SCUD launchers as an application, there is no reason why this algorithm can't be applied to other objects. For that reason, instead of saying mobile SCUD throughout the paper, we will refer to what is being observed as the object.

Scope of the Study

In considering how to predict where a given object will be at some point in the future, the definition of the problem is critical because there are infinite ways to approach the problem. The design has to reflect the desired outcome and initial inputs. The desired outcome is information made up of bearing sequence predictions and expected protocols. Movement can be inferred from that information depending on the application. The observational data will come in the form of direction bearings. Combined with terrain data, these initial inputs will parameterize the model.

This problem will begin with all the given terrain information assumed. The includes road maps, weather, moisture gradients, vegetation, etc. This is significant because a lot of work goes into getting this information. SAR imagery, road maps, and terrain maps can all be attained by satellite. The scope of this study does not include the positioning of these satellites at the proper time, remote sensing any of the images to a specific degree of accuracy, or processing these images into usable vector fields. This study will start with the terrain information and proceed [3].

An important assumption is that all the prior activities can be done correctly prior to this analysis. Operationally speaking, there are three different kinds of problems that can occur: modeling error, process noise, and measurement noise. The first error happens when the model order is incorrect. Process noise comes from the fact that even if the model order is correct, the detail is not the same as the actual system. Lastly, measurement noise comes from the instruments or the environment not being conducive for accurate measurement [10]. Assuming that the information given is correct nullifies process and measurement noise. Reducing these uncertainties is beyond the scope of this paper so we assume them to be nonexistent. This algorithm will try to select the model order and, therefore, reduce the modeling error. That will lend itself to better predictions of future movements.

After analysis, what is the final product? Once enough observations of the object are taken, the expected Markov state at any time, parameters of the model, and predictions for future movements will be the output information. The Markov state of the object will correspond to what state of mind the object is in, or what protocols it must follow. The parameters of the model will include how the object transitions between Markov states and what the probability of seeing a specific movement in a specific state is.

Finally, the prediction of future movements will be based on the parameters of the model and will show trends of where the object is most likely to move next.

The intent is to present an algorithm to model behavior of an object with HMM. As a result, no actual data has been used. All observations are a consequence of a random number generator. The observational data used for validation was designed and can be seen as part of Appendix I. There is no guarantee that the object will behave operationally in a way that lets us draw meaningful information from their movements in the same fashion that we will with the generated data, but that is an unavoidable uncertainty.

In order for this model to approach absolute certainty, infinitely many observations have to be taken over time. In an operational situation this might not always be possible. That further limits the ability of a position prediction and order determination for the object. In other words, the extent search areas may be narrowed is totally dependent on the observation time allowed.

Furthermore, only predictions of object bearing are given. In order to really predict future movements a velocity and time interval between observations would have to be introduced. This is dependent on the characteristics of the object and the type of orbit used for observation, respectively.

Trending future movements is not an exact science. Highest probable sequences will be given based on observed data and it will be up to the observer to draw actual conclusions about bearing. It will be simple in most cases but that is entirely dependent on the nature of the data and how much time the object is observed.

Understanding the mathematics behind this formulation is important because it gives insight into what can actually be done with the model. The methodology of the algorithm is the most important part of this study because of the new application of HMM. Knowledge of the methodology is necessary to make changes to the algorithm further along in the paper. Finally, algorithm validation and possible future uses of the formulation will be presented.

CHAPTER II

MATHEMATICS BACKGROUND

An understanding of some basic math concepts behind the hidden Markov model is essential to understanding the definition and formulation of the problem.

Stochastic Processes

All of the modeling done in the paper can be understood within the confines of a stochastic process. Generally speaking, a process is stochastic if for every λ event in the sample event space S a real-valued time function $X(t,\lambda)$ is assigned. For each event, λ, there is a single time function $X(t,\lambda_i)=x_i(t)$ called the sample function. The totality of sample functions for a system is called an ensemble. For a fixed time t_j, $X(t_j,\lambda)=X_j$ is a random variable [11]. Figure 1 illustrates the notation. Notice that the initial conditions of the sample functions are different for each sample function. Also, the initial conditions do not dictate the entire sample function.

Stochastic processes can also be defined as a collection of random variables $\{X_t\}$, where t runs through a given index T. Often the set T is taken in time so it is usually non-negative integers [12]. X_t is some measurable characteristic at each t in the index T. For a fixed t and λ, $X(t_i,\lambda_i)=x_i(t_i)$ will become a number [11].

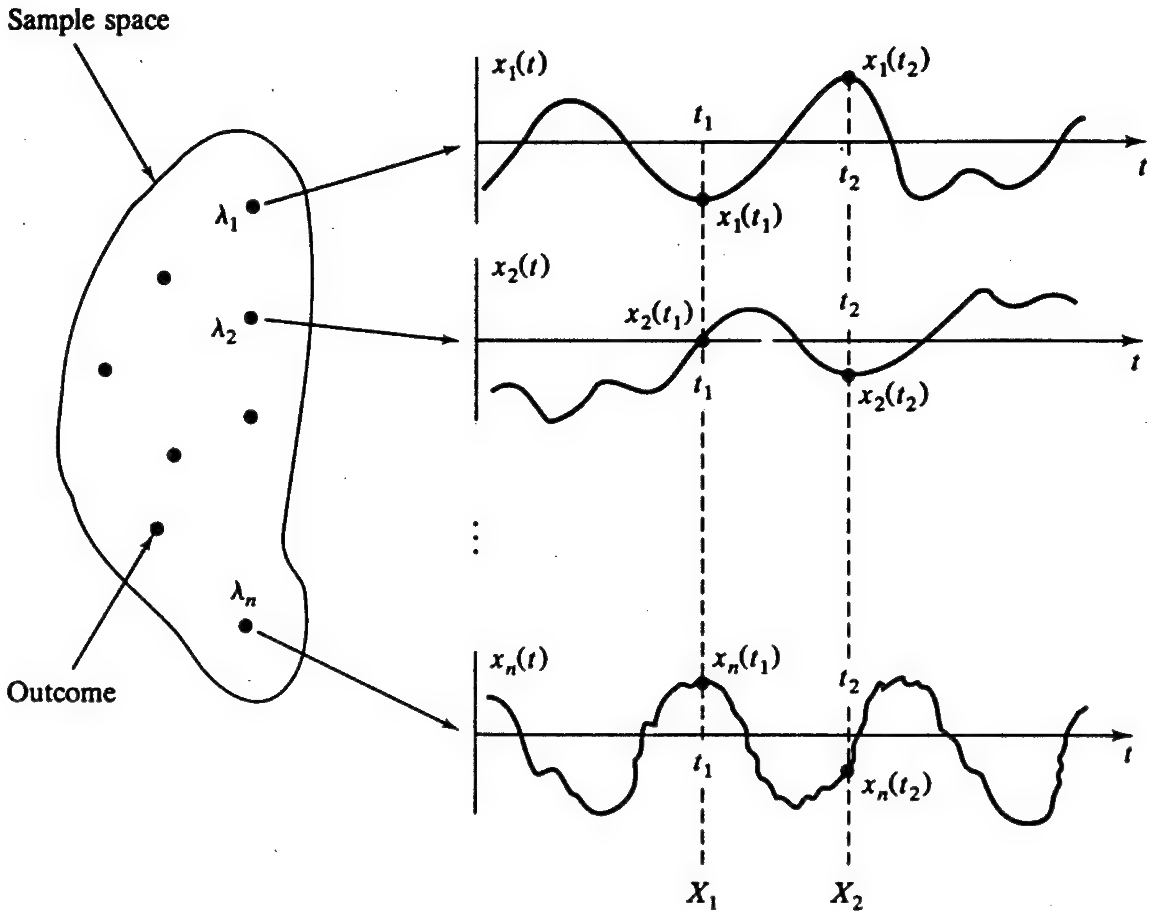


Figure 1. Random Process
(Extracted from [11], pg. 184)

An example of a stochastic process is the toss of a coin or the measurement of the temperature each hour of the day. In the case of the coin toss, an event λ, might be getting three heads in a row. The sample function would look like three heads in a row. The random variable X_j is the side of the coin shown at the particular time t consistent with the sample space. Care must be taken when defining stochastic processes because probabilities are assigned to events and not outcomes. In the case of tracking object movements, the random variable can be thought of as the measurement of a object's vector position at any time t consistent with the definition of the sample space. The event space would include all possible bearing sequences up to a prescribed time t.

A numerical outcome whose value can change from experiment to experiment can be thought of as a random variable. The formal definition is a real valued function whose domain is a sample space [13]. From the temperature example, the sample space would include all the possible sequences of temperatures for that season up to a time t. The real valued time function $X(t,\lambda)$ would be the measurement of those temperatures sequences. At each time step the random variable temperature can be measured from the available temperature range for that season.

The random variables are assumed discrete for this study. The random variable X is said to be discrete if it can take only a finite number of possible values of X [13].

For the methodology, an outcome of the model is assumed to be a multiple of 90 degrees. More complexity requires the outcome multiple to be smaller.

Stochastic processes can vary in the form they take on but for this study, at particular points of time t labeled 0, 1, ..., the model is found in one Markov state.[2] These states are mutually exclusive and exhaustive and are labeled 0, 1, ..., M.[3] The points in time are spaced depending on the behavior of the physical system that the process is embedded in [12]. For example, a measurement can be taken at a change in the system or at regularly spaced intervals. For our purposes, measuring a subject's location at spaced intervals is most appropriate because a satellite will pass overhead in certain time intervals.

Markov Processes

There are many stochastic processes that are of interest. A Markov process is just a stochastic process that exhibits the Markovian property:

$$P\{X_{t+1}=j \mid X_0=k_0, X_1=k_1, \ldots, X_{t-1}, X_t=i\} = P\{X_{t+1}=j \mid X_t=i\} \tag{1}$$

This is true for time t = 0,1,... and every sequence k_0, k_1,..., k_{t-1}, i, j. This is one of the assumptions that allow stochastic processes analytical tractability. Other

[2]There are only a finite number of states to the model.

[3]Mutually exclusive means that the elements of a set in a sample space do not overlap. Exhaustive means that the list contains all possible outcomes [13].

assumptions about the joint distribution of X_0, X_1,... are also necessary and will discussed later in the paper [12].[4]

Equation (1) basically says that the probability of a future event only depends on the present state. The conditional probability that the system will visit any other state is independent of all states except the present state [12,13].[5] This is important because all of the other states that the model has visited are not necessary for future predictions. Essentially this assumption makes the model simpler to define.

In the model dealing with location prediction the Markov states will correspond to the protocols and states of mind the launch team has to operate on. In other words, the Markov state will correspond to the rules they have been given and what frame of mind they are in. Different states can be designed to mimic different possible sets of orders. For this case, the Markovian property is assumed. The crews operate solely on their current orders and what they perceive danger to be at the present time. Once their Markov state changes (i.e., their orders or perception of danger) they will move according to that new state exclusively.

[4]Let X_1 and X_2 be discrete random variables. The joint probability distribution of X_1 and X_2 is given by $p(x_1,x_2) = P(X_1 = x_1, X_2 = x_2)$ [13].
[5]If A and B are any two events, then the conditional probability of A given B is given by $P(A \mid B) = P(AB)/P(B)$. This is provided that $P(B)>0$. Additionally, if two events are independent then $P(A \mid B) = P(A)$ [13].

The notation for the probability of going from one state to another is:

$$p_{ij}^{(n)} \tag{2}$$

It is the conditional probability that the random variable X, starting at state i will be in state j after exactly n steps [12]. By definition, the conditional probabilities have to be non-negative. They have to obey the following rules:

$$p_{ij}^{(n)} \geq 0 \qquad \text{for all i and j; n=0,1,2...} \tag{3}$$

$$\sum_{j=0}^{M} p_{ij}^{(n)} = 1 \qquad \text{for all i; n=0,1,2,...} \tag{4}$$

These rules spring from the fact that the system must make a transition into some other Markov state [12].

Ergodic Systems

The ensemble was defined above as all the sample functions grouped together. It is often useful to describe these groups of sample functions in the context of statistical averages. Consider the stochastic process X(t). At time t_1, $X(t_1)=X_1$ is a random variable with distribution:

$$F_X(x_1;t_1) = P\{X(t_1) \leq x_1\} \tag{5}$$

where x_1 is any real number. The first order density function is:

$$f_X(x_1;t_1) = \frac{\partial F_X(x_1;t_1)}{\partial x_1} \tag{6}$$

The mean of X(t) can be found by:

$$\mu_X(t_1) = E\left[X(t_1)\right] = \int_{-\infty}^{\infty} x f_X(x_1;t_1)dx \tag{7}$$

If the stochastic process is wide-sense stationary, the mean is constant in time [11].

$$E[X(t)] = \mu_x \tag{8}$$

The time-averaged mean of a sample function x(t) of a stochastic process X(t) is defined as:

$$\overline{x} = \langle x(t) \rangle = \lim_{T\to\infty} \frac{1}{T} \int_{-\frac{T}{2}}^{\frac{T}{2}} x(t)dt \tag{9}$$

By being stationary, the expected value of both sides of equation (9) can be taken. It will yield [11]:

$$E[\bar{x}] = \lim_{T\to\infty} \frac{1}{T} \int_{-\frac{T}{2}}^{\frac{T}{2}} E[x(t)]dt = \mu_x \tag{10}$$

If the result from equation (10) is assumed to be true, the time averaged mean from equation (9) and the ensemble mean from equation (7) are equal. By definition then, an ergodic process is one that has equal time averages for each sample function and are also equal to the corresponding ensemble averages [11]. The key to a process being ergodic is to be wide-sense stationary and have statistics that do not evolve with time as per equation (8).

This result may seem obscure but has broad implications. For example, if a process is ergodic, then by observing a single sample function all the relevant statistics can be determined without observing more sample functions [11]. In other words, observing the process during any period of time will allow a full description of that process throughout time. This process is assumed to be ergodic.

Hidden Markov model

The hidden Markov model is the next step in the complexity of the Markov process. In a Markov process, the shift from one Markov state to another is probabilistic while the output is deterministic. The hidden Markov model extends this simple chain to a doubly stochastic process. In other words, both the output symbol and Markov state transition are determined probabilistically. The sequence of the state transitions is "hidden" because the observer only physically sees the symbol outputs [4],[14],[15],[16].

An example of a simple Markov process is a coin toss. The process is given two states. The states are represented by a heads or tails [14]. Each state will output either a heads or tails as the observation symbol, respectively. The output symbol has no probability distribution for a given state. A hidden Markov model of the coin toss, on the other hand, could also have two Markov states. These states, however, are represented by differently biased coins. Each coin will turn up either heads or tails based on a different probability distribution. The output of each state now has a probability distribution associated with it. Each state can output either a head ot tail. The process has become doubly stochastic. Modeling coin tosses this way could be important because the more complex hidden Markov model could be better suited to modeling outputs.

Because the hidden Markov model (HMM) is a doubly stochastic process, it is more difficult to define. Let N be the number of Markov states in the model. The individual Markov states will be denoted $S=\{S_1, S_2, ..., S_N\}$. The number of distinct observation symbols per Markov state is given by M. The individual symbols are denoted $V=\{v_1,$

$v_2, \ldots, v_M\}$. The state transition probability distribution $A=\{a_{ij}\}$ is given by:

$$a_{ij} = P[q_{t+1} = S_j | q_t = S_i] \quad ; 1 \le i \text{ and } j \le N \qquad (11)$$

The observation symbol probability distribution in state j, $B=\{b_j(k)\}$ is given by:

$$b_i(k) = P[v_k(t) | q_t = S_j] \quad ; 1 \le j \le N \text{ and } 1 \le k \le M \qquad (12)$$

Finally, the initial state distribution $\pi = \{\pi_i\}$ is given by:

$$\pi_i = P[q_1 = S_i] \quad ; 1 \le i \le N \qquad (13)$$

With the parameters of N,M,A,B, and π the HMM can generate a sequence of observations $O = O_1, O_2, \ldots O_T$ [14],[16].

Given these parameters, several problems with real-life applications can be solved. The most basic problem involves finding an efficient way to compute $P(O|\lambda)$.[6] This is simply the probability of observing a sequence of events given the model parameters [14]. The highest probability sequence should indicate where something will move in the future. Examining several of the highest probability sequences, trends should be able to be drawn as to where objects will be later in time.

[6] λ denotes the parameters of the model A, B, and π

The second problem involves finding a state sequence, Q = $q_1, q_2, \ldots, q_T$, that best explains the observations taken. This commonly involves a dynamic programming technique called the Viterbi algorithm [17]. Although no totally correct sequence can be found, it does show most probable sequences and can lend insight into the model [14]. By knowing what state the teams are in, we can lend further insight as to where the object may be based on what we think their protocols are.

Finally, the third problem involves re-estimating the parameters. This is done in order to maximize $P(O|\lambda)$. In the end, much of the modeling would be worthless if there was no way to improve on how we estimate what will happen in the future [14].

As practical as this type of modeling seems, it does have setbacks. Numerically, as time goes to infinity underflow problems for the probabilities become more of an issue. Ironically, a infinite number of samples is required to generate a perfect set of parameters for the model [14],[16]. Secondly, the necessity for finite training sets forces some observations to 0 probability as the re-estimation procedure alters the observation matrix [14],[16]. Finally, the re-estimation procedure tends toward the local maximum of a likelihood function. The likelihood function can be complex with many local maximums so, estimating initial parameters is important when trying to reach a global maximum and speed up run time [14]. All of these problems will be addressed later in the methodology of the problem.

Theoretically, the Markovian property itself sometimes oversimplifies the nature of the model. Only being dependent on the current state is sometimes too big of a simplifying assumption. Also, HMM is only well defined for single independent variable function. This is much more of a problem than the current state assumption because it is currently impossible to define a HMM for more than one independent variable. In principle, however, HMM can be made dependent on more than its current state [4].

CHAPTER III

METHODOLOGY

Referring back to chapter II, the HMM needs some basic definitions in order to solve the problems mentioned. The λ parameters need to be defined, with the number of Markov states, and the number of observation symbols allowed. Also, the sequence of events needed to reach the end result has to be presented.

The first operation after estimating the initial parameters and gathering observations consists of finding the forward and backward probabilities. These probabilities help to find probabilities of sequences and re-estimate parameters of the model.

Once the final parameters of the model have been found through re-estimation, the probability of the most likely sequences can be found. This can then be compared to what an ideal case would look like. The particular model can be evaluated next to several other models to determine the best order. Once the final model is selected, trends can be seen as to where the object will go and what its protocols are.

Define the Model

The order of the model can be chosen at random. The definition of the model includes the number of states, N, and the number of observations allowed, M. The methodology shall start with a fundamental model and work to more complex models to see if they model behavior any better. To help understand the significance of the order it is useful to make M = 4 and N = 3, initially. Later on, there will be different orders to consider but this initial model is simplest to explain and understand in terms of observables and Markov states visited. The 4 observations possible correspond to the 4 points of the compass (North, East, South, West). This will give look angles of 90 degrees around each direction of the compass. In the MathCad formulation, the directions of the compass will be represented as 0, 1, 2, and 3, respectively. The 3 states allowed correspond to the state of mind of the team driving the launcher and/or the rules of engagement they must follow.

One state represents trying to be expedient. The team might have orders to get out of the area as soon as possible or get to a launch site by a certain time. Another state represents trying to be concealed. Orders might include hiding in the local area or evading incoming warplanes. The last state represents one of ambivalence. Lacking all instruction and information, traveling in any direction will be equally likely.

These definitions are, of course, not the only way to define models. The number of observations we allow is only limited by computer speed. The included look area that each observation has will only decrease with more observations allowed. The number of states allowed is limited by computer speed and creativity. Other than what was mentioned above, states can be also defined by moving versus stationary orders, pre-launch versus post-launch status, or any combination thereof. Through our testing with MathCad, the number of states was the most significant factor in determining run time on the computer. The scope of this study will focus on 4 and 6 observations with 3, 4, and 5 states allowed. Adding states by re-formulating the model will be an exercise to examine the order of the model and will be estimated with a beta density function.

Estimate initial matrices

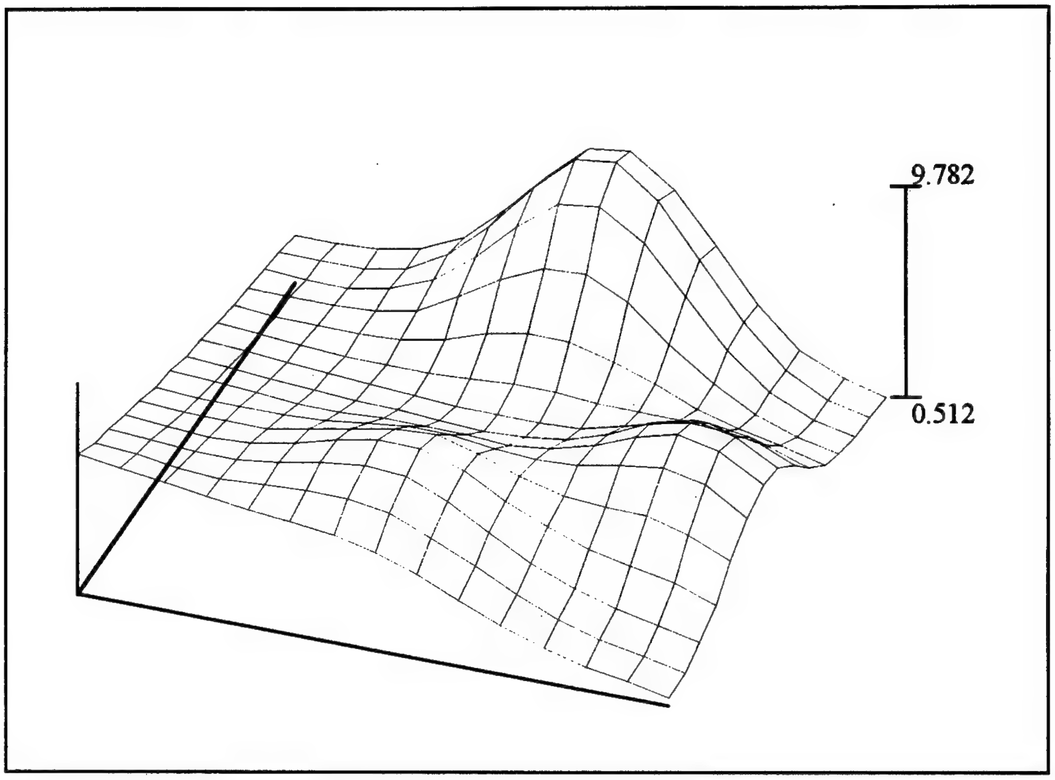
In order to begin the simulation an initial model has to be defined. These matrices represent apriori probabilities of the model. With these probabilities, the model can be started close to what the optimal model is. The better the algorithm is in estimating the matrices, the faster the re-estimation will be.

The most difficult matrix to estimate was the observation probability matrix, B. The MathCad algorithm can be seen as part of Appendix A. With all terrain, weather, and road maps at hand, the individual vector fields can be aggregated to form one final field. As mentioned above, this operation is in the realm of image processing and assumed given. Therefore, we can estimate an aggregate field right away and proceed with the algorithm. A possible example of this aggregate can be seen as part of Figure 2. It is a relative plot to show what the vector field might look like. Units and components will be specified by the application. It is assumed that the higher sections of the z-function correspond to poor weather conditions, bad roads, or good concealment. Conversely, launch minded expedient teams will tend to move towards the lower sections of the z-function. This convention, however, will also be chosen to fit the application.

Another convention chosen was to begin the object in the middle of the aggregate field. Then by differencing that element with every other, the object could start at a relative ground zero. The values of the new vector field can be seen as part of matrix M on the page following the contour plot in Appendix A.

The differencing above poses a problem. The M matrix can have negative elements if the middle element is higher valued than other elements. Negative numbers are not acceptable because these will translate later into probabilities that would be infeasible. Here, this problem was remedied by forcing a small positive number on any negative sum. Of course, problems can arise from such a simple approach (i.e., starting at the top of a hill) but, it will suit fine for most applications.

Now, by looking in each direction the maximum value can be determined by looking at the associated look angle. Adding the maximum z-function values together in a weighted



M

Figure 2. 3-D Plot of Aggregate Fields

sum across the look area will translate into the apriori probabilities for that direction.

These algorithms can be seen as part of Appendix A. They are named according to the direction in which they look. The bars associated with each direction denote MathCad's programming structures. The east program, for example, finds the maximum z-function value of the matrix M looking in the east direction. Care must be taken in deciding where to look because the top of the matrix doesn't correspond to north on the map from figure 2. The 0,0 element on figure 2 is at the lower left corner while the 0,0 element on the matrix is in the upper left.

For the east direction, look to the top of the M matrix. Starting at the right and working left in columns[7], each applicable row is examined and the highest value is stored[8]. A while loop is given to each column in the look area for that direction. A weighting is given to each column according to how directly it faces the corresponding direction. The column directly east is weighted the highest while columns on the fringe of the look area are given a smaller weight. The west direction is the same as east but will look down the matrix M instead. With the other two directions, the rows will take the place of columns and vice versa. As the look areas become smaller due to more allowable observations, the number of while loops will

[7]The columns are denoted by the local variable b in these programming structures

[8]A row is considered applicable if it falls within that direction. A row towards the bottom of the matrix would not have anything to do with being in the east direction.

decrease while the number of total programming structures will have to increase.

Dividing each direction by the sum of all directions will normalize the sum to unity. Negative numbers in this operation would cause meaningless probabilities. Depending on how the states are defined for a particular model, these numbers can be manipulated to represent the probable directions for that state. The straight forward calculation above makes for acceptable estimates of the likelihood of moving in a particular direction for a conceal state, for example. This is true because of the way the convention was arranged earlier. If, on the other hand, the object needed to move quickly, staying on the low portions of the vector field would be desirable. This can be described by taking the reciprocals of the direction numbers and adding them together the same way to get unity. For a complete description of how each Markov state calculated its initial observational symbol likelihood's see chapter IV dealing with these descriptions.

Whatever the formulation, the rows of the B matrix have to equal the number of states allowed and the summation of each row has to be unity as explained above. The number of columns of the B matrix has to equal the number of observation symbols allowed.

The state transition matrix, A, is not part of the physical data. This matrix shows how the object in question transitions between operating modes (i.e., Markov states).

The formulation will initially assume that the object will most likely transition back to the state it left from in one step. In other words, the changes in state typically won't occur because the Markov state is relatively stable over time.

The beta density function will be used to estimate the transitions from state to state of the HMM. The beta density function is convenient to use because it is easy to parameterize to allow for recurrence and it terminates at unity. Segmenting the density function according to the number of states and integrating will give starting points for the A matrix that make physical sense in accordance with the assumption above.

Appendix A experiments with several different methods for representing the matrices. They serve as a check on the most general form. The diagonal elements of the A matrix in the Appendix have the highest probabilities which can be interpreted as a good chance of recurrence in a single step. This matrix is square with each side numbering to N, the number of allowed states. The number of observations don't contribute to the definition of this matrix because it deals strictly with the states of the model.

The initial state matrix can also be figured by beta density function integration. In the case of the 3 state model, starting off in the ambivalent state is assumed so the beta distribution is parameterized to allow for that. This matrix has one row and as many columns as allowed

Markov states. The row, as always, has to sum to unity to be axiomatically correct.

Observational data

The aggregate vector fields are not the only physical data that needs to be collected. Observing the bearing vectors is necessary to re-parameterize the HMM. Here it will be done by a random number generator. Appendix B shows the MathCad for generating observations. The number of times to re-estimate the matrices is represented by the number of columns in the observation matrix O. The re-estimation algorithm executes after each set of data is seen. The sets of data correspond to the columns of the O matrix. Thus, by controlling how many columns there are in the matrix and the length of the columns, we can decide how often to re-estimate the model parameters.

By asking for random numbers based on a specific distribution there is an added advantage. This is essentially like taking the observations from a true matrix biased to a specified state. The bias state can be determined by how we parameterize the random number generator. Essentially, we will see if the formulation is stable later on by giving it a designed matrix and seeing if the model re-estimates to an obvious state. If the most likely state is similar to the true state designated by the designed data, the model is re-estimating well. These tests are included in the results chapter.

The O matrix in Appendix B can be interpreted as the data that the satellite gathers as information before a prediction has to be made on future location. It can be thought of as data that is from the distant past. For models with 4 observations and different numbers of states the O matrix must not change. Furthermore, models with 6 observation symbols and different states have their own set of data because the definition of the look area changes and becomes smaller. This was done because one data set could have data from 0-3, while the other data set could range from 0-5. The sets were kept the same across the state variations so each formulation could be re-estimated equally. This allows for good comparisons between models later.

The data that will be used to do the comparing is the realO matrix also in Appendix B. This data can be thought of as being in the recent past. This matrix also varies between models with different observations symbols. Like the O matrix, however, it is the same across the number of allowed states. Once the models have been parameterized with the O matrix to their final λ parameters, the realO matrix will be used in comparing the models with different states.[9,10] The probability of the entire realO matrix will be assessed with the final λ parameters. The model that

[9]The final matrices can be thought of as the re-estimated matrices used at the time when a prediction is needed.

[10]Remember that λ denotes the matrices A, B, and π in the HMM formulation.

predicts this realO with the highest probability will be considered the proper model to minimize modeling error. The number of columns and rows can change from the realO to the O matrix to demonstrate some versatility in the model formulation. The comparison between order of models can be seen as part of the results chapter.

Forward and Backward Probabilities

In order to solve for the probability of certain sequences and all the other problems associated with HMM, the forward and backward probabilities have to be solved. These variables come from the fact that a complete enumeration of probabilities for each sequence of outcomes is computational infeasible. The calculation is on the order of $2T*N^T$ [14], where T is time observed, while N is the number of states. With larger values of N and T the problem quickly expands exponentially.

The forward variable, $\alpha_t(i)$, is defined as the probability of a partial observation sequence, O_1, $O_2, \ldots, O_T$ and state S_i at time t, given the model λ. The key to this concept is that since there are only a certain number of states, regardless of how long the sequence is, the object is constrained to transition among the allowed number of states throughout the sequence. The definition the forward variable is:

$$\alpha_t(i) = P(O_1, O_2, \ldots, O_t, q_t = S_i | \lambda) \tag{14}$$

To solve for the forward variable, two steps have to be taken. The first step includes finding the joint probability of being in state S_i and finding observation O_1.

$$\alpha_1(i) = \pi_i b_i(O_1) \tag{15}$$

The i index is iterated from 1 to N. The second step is represented by Figure 3. It shows how to get to state S_j from S_i at any time t+1. This step inductively finds every forward variable after the initial.

$$\alpha_{t+1}(j) = \left[\sum_{i=1}^{N} \alpha_t(i) a_{ij}\right] \cdot b_j(O_{t+1}) \tag{16}$$

The j index is iterated from 1 to N, while the t index is iterated from 1 to T-1. Summing $\alpha_t(i)a_{ij}$ over every state yields the probability of being in state S_j at time t+1. This is true because $\alpha_t a_{ij}$ is the joint probability of seeing O_1, $O_2, \ldots,$ O_T and S_j at time t+1 from state S_i at time t. The calculation effort here is on the order of N^2T. This saves orders of magnitude in the number of calculations required [14].

The backward variable is essentially the mirror image of the forward variable. It is the probability of the partial sequence of observations from t+1 to T, given state S_i, at time t and model λ. The backward variable can be

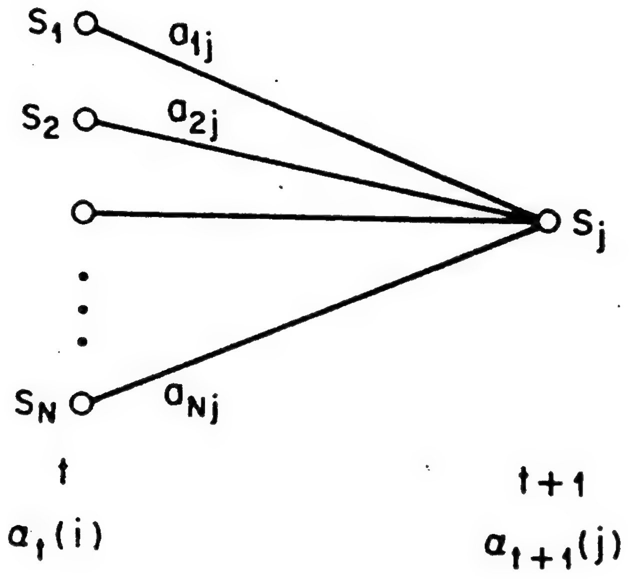


Figure 3. Representation of Computations required for Forward Variable (Extracted from [14], pg. 262)

given by:

$$\beta_t(i) = P(O_{t+1}, O_{t+2}, \ldots, O_T | q_t = S_i, \lambda) \qquad (17)$$

Since it works backward, the initialization step has to set the $\beta_T(i)$ equal to unity for states 1 to N as in equation (18).

$$\beta_T(i) = 1 \qquad (18)$$

The induction step can be seen in Figure 4. It shows that in order to be at state S_i at time t and to account for the observation sequence from time t+1 to T we have to account for being in all states S_j at time t. The induction step can be seen as equation (19).

$$\beta_t(i) = \sum_{j=1}^{N} a_{ij} b_j(O_{t+1}) \beta_{t+1}(j) \qquad (19)$$

The a_{ij} term above accounts for the transition from S_i to S_j. The $b_j(O_{t+1})$ term accounts for the observation seen at time t+1 in state j. Finally, the $\beta_{t+1}(j)$ term accounts for the rest of the observation sequence from state j. The time step starts at T-1 and works to 1 for all states from 1 to N. The computational effort is also simplified here along the same lines as above in the forward variable [14].

The forward and backward variable are not numerically simple to compute. It is not difficult to see that for large values of time the forward and backward variables will result in underflow of any computer. Even the small scale model shown in the methodology results in very small probabilities. Of course, the underflowed values will be at

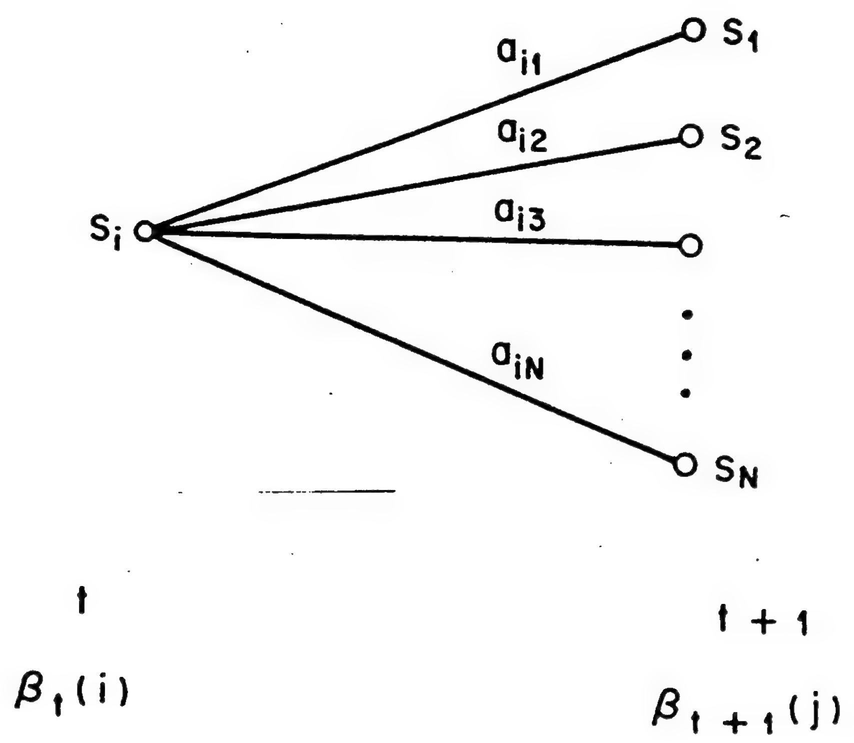


Figure 4. Representation of Computations required for Backward Variable (Extracted from [14], pg. 263)

T for the forward variable while the problem for the backward variable arises at the beginning of time.

The actual matrices for this model can be seen as part of Appendix C under FwdProb and BackProb, respectively. The solution to this problem comes from scaling the variables to stay within the dynamic range of the computer. Essentially, the scale is the inverse of the forward variable summed for the number of states in the model [14],[16].

The scaling factors were calculated by two ways in Appendix C. The first way was done in traditional MathCad symbology as the (scale) variable. The second way was accomplished using the programming structures within MathCad and can be seen as simply (c) in Appendix C. Both ways were done to assure the procedure was correct. Improper scaling could seriously flaw the reliability of final results.

Once we were assured that the numbers were correct, they could be used in calculations. By multiplying the scale by each forward and backward variable throughout calculations they will cancel each other out by divisions. It is a convenient way to deal with the underflow problem and can help us to calculate sequence probabilities later on [14],[16].

Most likely State

Knowing what Markov state the model is in at a particular time is difficult because it is not seen. To recall the bias coin example, it would be like finding out what bias coin is being used based on only a set of heads and tails. For the tracking problem being discussed here, the bearing observations will provide the only clue as to what protocols the object is using.

The mathematics behind determining this can be very complex. The Viterbi algorithm is one way to find the best sequence of states for any given set of observations. It maximizes the probability of the single state sequence given the observations and the parameters of the model (max $P(Q|O, \lambda)$ [14],[17]. Although there are plenty of HMM applications that use the above algorithm, the most probable sequence is of little use in this application. The current state is more important for this situation. That way trends for future movement can be verified against what the believed protocols are at that time.

In order to find out the most likely state at each time, we want to maximize the expected number of correct individual states over a time period. To do this, we need to find the probability of being in state S_i at time t, given the sequence O and the model λ.

The formal definition is [14]:

$$\gamma_t(i) = P(q_t = S_i | O, \lambda) \tag{20}$$

Appendix C shows how it can be defined in terms of forward and backward variables. It is a probability measure so summing over states for each time yields unity. Verifying this in the calculations indicates the mathematics are sound up to this point. Errors in definition and indices would yield results other than unity for the summation.

The measure serves to show the most likely state at a particular time. The problem with the calculation is that it doesn't have any regard for the probability of occurrence of sequences of states. In other words, it could solve for a bogus sequence of states that aren't even possible because the state transitions aren't feasibly allowed [14]. As mentioned above, the current state is the most important information, while the sequence is of little use. This allows us to use the criteria for most likely expected state at individual times rather than expected state sequence of the Viterbi algorithm.

As an additional check on the scaling factors, see both calculations of γ in Appendix C. With and without the scaling factors the matrices should be equal. Since they both are, the calculations are correct. To actually determine the most likely state, look down each column (time) and across each row (Markov state) in Appendix C. For each time (column) there will be a row (state) with the highest probability. This will be the state that the object is expected in and the number will show what that corresponding probability is. Obviously, if there is not a state that has a high probability it will be hard to draw useful information from the γ calculation in equation (20).

Re-estimation of Parameters

The re-estimation of parameters is the most tedious and difficult problem to solve in the study. It is, however, the most important. Without being able to modify the parameters of the model, observations would be useless. Any predictions of movement would have to be based on the initial parameters estimated before any data was taken.[11] This defeats the purpose of observing in the first place.

The algorithm for re-estimating can be seen as part of Appendix C in the matrix routine. Yet, another variable can be defined to help validate the formulation. The probability of being in state S_i at time t, and state S_j at time t+1, given the model and observation sequence is defined as:

$$\xi_t(i,j) = P(q_t = S_i, q_{t+1} = S_j | O, \lambda) \tag{21}$$

Writing this equation in terms of forward and backward variables is part of Appendix C. Again, if all formulation and mathematics work has been done correctly up to this point then:

$$\gamma_t(i) = \sum_{j=1}^{N} \xi_t(i,j) \tag{22}$$

For the time 0 of the model shown in the calculations of Appendix C, the left hand and right hand side of equation (22) equal each other. A slight change of notation occurs

[11]Remember that the parameters of the model (λ) are given by the matrices A, B, π.

here. The ξ of equation (21) is represented by a Ξatzero in the appendix. It will become clear as to why later on.

The matrix algorithm looks more complicated than it is. All the small routines are set up within one large while loop. This while loop is conditioned on being less than K to operate. Recall from Appendix B that K is defined as the number of columns in the observational data (O matrix). Also, recall that the observations are set up in a matrix so a re-estimation can take place after each column. The while loop here is what causes a re-estimation every column.

Inside the main loop several variables are initialized. The next major step in the algorithm is the calculation of Ξ. The definition of Ξ is:

$$\Xi_{i,j} = \sum_{t=1}^{T-1} \xi_t(i,j) \qquad (23)$$

This is the expected number of transitions from S_i to S_j [14]. Ξ has time numbered columns and state numbered rows. This calculation is a triple nested loop for every element of the matrix and every time.[12] Trying to build the ξ matrix of equation (21) is difficult because it has essentially three dimensions. This poses an implementation problem for MathCad because it will not recognize such a matrix as valid. Therefore, the Ξ matrix had to be constructed at each iteration of the Ξ loop. The sumZeta notation simply handles adding the appropriate time elements together.

[12]The above Ξatzero was defined that way because it was at the first time possible. When dealing with only one time the summation can disappear and the Ξ and ξ variables equal each other.

Notice how the variables for each separate loop are re-initialized at the end of the loop rather than at the beginning.

Once the Ξ matrix has been constructed, the γ matrix can be calculated for the while loop. This is the same variable as defined before in equation (20). Now that we are in a loop, this value has to be kept current with the λ at the corresponding time. The previous work was for purposes of seeing what was going on qualitatively and making sure the mathematics were constructed properly. The γ matrix will be indirectly calculated from λ. Therefore, using the λ from previous work (i.e., outside the loop) would yield incorrect results.

With the two above matrices constructed, a re-estimated A_{ij} matrix can be determined. The operation is given by:

$$\overline{A}_{i,j} = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)} \qquad (24)$$

The new A matrix requires summing over i and j as seen in equation (24). Essentially, the new A matrix is the expected number of transitions from state S_i to state S_j divided by the expected number of transitions from state S_i [14].

The new π matrix is almost trivial. It is taken simply as:

$$\overline{\pi}_i = \gamma_1(i) \tag{25}$$

It is the expected frequency in state S_i at time t=1 [14].

The new B matrix is much more complicated. It will require the operation:

$$\overline{B}_{i,j} = \frac{\sum\limits_{\substack{t=1 \\ s.t.O_t=v_k}}^{T} \gamma_t(i)}{\sum\limits_{t=1}^{T} \gamma_t(i)} \tag{26}$$

This variable is essentially the expected number of times in state j and observing symbol v_k divided by the expected number of times in state j [14]. First, notice the index on the time. It is from T rather than t-1. This is because the B matrix handles observations which occur at every time. The last state transition occurs at t-1. Because the A matrix handles state transitions, its index can only go as high as t-1. Secondly, notice the note under the top summation. This summation is conditioned on the symbol observed. In MathCad, the Y variable handles this summation. The column of Y to get the γ added to it is the same column corresponding to the observation symbol seen. The observed symbol, of course, comes from the training data O matrix. The element depends on where K is and at what time the loop is looking at.

Another implementation issue arises in this algorithm, however. Because no data set can ever be complete, the re-estimation algorithm can conceivably create elements of a B matrix that are equal to zero. If the training set never sees one symbol it will re-estimate that symbol probability to 0. That is never really true so a default value (ε) has to be given to that matrix element in the absence of training data. Once that is done, the row has to be rescaled so that the sum of the row is equal to unity [14].

Once all the matrices have been re-estimated, they have to be reset to the old matrices so the loop can proceed with the new parameters. Once the final column of training data is processed, the final re-estimates of the parameters become the output matrices. MathCad will only output one matrix at a time so all three matrices are combined for output. Once out, the aggregate matrix is separated into its components again.

Of course, all of the axioms of probability must be obeyed to have meaningful parameters of the model. All of the rows in each of the matrices have to be equal to unity when summed. This check will serve as a test of the algorithm's correctness. It has been proven that using the procedure in Appendix C will either generate new A, B, and π matrices that increase maximum likelihood of a sequence or stay at an already critical point in the likelihood function [14].

Determining the best observation sequences

Once the parameters of the model have been re-parameterized finally, the highest probability observation sequences may be determined. The probability of a given observation sequence is given by:

$$P(O|\lambda) = \sum_{i=i}^{N} \alpha_T(i) \tag{27}$$

These probabilities can get extremely small for long time durations. In that case, using log probabilities will be best because computer underflow becomes an issue. Fortunately, the time durations used don't cause these problems so the calculation can be direct.

Time durations are kept small because of the necessity for this problem to come up with an answer in a short time. If the algorithm took days and days of observation to analyze the object, it would probably not be any better than conventionally searching and destroying the target.

The essence of this new problem revolves around finding the probabilities of the most likely observation sequences. Looking at equation (27) again, the forward variable calculation has to be accomplished first. Appendix D starts out with this exact calculation. Notice that the scale is not included.

After that, the probability of each sequence has to be determined. For very small models this can probably be done manually although it would be very tedious. For any model the possible number of combinations of symbols is given by:

$$n^{r} \qquad (28)$$

(n) denotes the number of possible observations. (r) denotes the length of the observation set [18]. For this model we have 4 possible observations and are allowing the set to be 3 symbols long. By doing the calculation, there are 64 different sets of observations to extract the probability from. For models of more complexity this number goes up exponentially.

The sequence routine in Appendix D generates all the possible sequences for our model. It requires as many nested loops as there are number of observations allowed in the length of the set. Because it allows the same number of sequences as equation (28) the procedure was done correctly.

The LOGP routine does the actual calculation of probabilities. It uses the forward probabilities generated at the beginning of the Appendix to output a probability for each sequence. The Bestset routine is a combination of both sequence and LOGP routines to generate a probability of each observation sequence. Part of the output can be seen below the routine. As we can see, the code starts to enumerate all the sequences and find the probability associated with each above it. The 0,0 element shows how many total sequences there are and the 0,1 element shows what the final summed probability across row 1 is.

This summed probability should sum to one. This is because we are looking at the entire event space. The reason why the sum isn't unity is numerical. Allowing the time to only go to 2 iterations will indeed sum the row to unity. Allowing the time to go to 4 iterations, however, will further degrade the summation. After much deliberation it was determined that because of the small nature of the probabilities and the sheer number of them needing to be added together, it is entirely possible to generate the error MathCad shows. This conclusion is consistent with allowing the time to vary and observing how the summation gets closer to and farther from unity.

For all practical purposes, the end result will not be affected. Relatively speaking, the sequence order will not change because all probabilities will be affected by the same issue. One way to deal with it, therefore, is to simply rescale all the probabilities. Dividing all probabilities by the sum of probabilities will yield numbers that will, in fact, sum to unity. The Bestset routine contains such a normalizing operation so that the final probabilities do sum to unity.

Once all the sequences have been identified and given a probability with Bestset, they are sorted out to find the most likely sequences. The rsort function in MathCad does that for us. It sorts the matrix from smallest to largest based on the first row of the column. Taking the transpose, reversing it, and taking another transpose was the easiest way to get the program to sort from largest to smallest. MathCad doesn't have a function that will sort from largest to smallest automatically. The rsort routine is based on a heapsort algorithm found in Numerical Recipes in C [19]. The new matrix can be seen below the manipulations. It sorts the most likely sequences from most probable to least probable.

Determining the best model and seeing trends

Once the most likely sequences have been sorted out, trends can be determined in how the object will move. An incorrect model will lead to poor information about the object, however. On the other hand, knowing the proper order will improve the quality of the information. This section is based primarily on trying to find the best model given the limited training sequences used to re-parameterize the model.

Up to this point, all observations have been from the observed data in the O matrix. This is distant past data that is considered given. The realO matrix is data that was seen in the recent past (see Appendix B). This division of the data is arbitrary and used to help verify the model order. It is the realO observations that will be used to determine what model is most appropriate for that data. It is important to realize that we should not use the same O matrix to determine the order. The O matrix was used to parameterize the model so the parameters reflect that matrix specifically. A new set of data has to be given to show best order most appropriately.

Unfortunately, when the number of observation symbols for a model changes, this matrix will have to be regenerated. This is because one model won't be able to handle the same number of symbols that another model might demand. The number of observation symbols for this application of HMM can be tailored to fit how much data is available and to what degree of accuracy we need. Computer speed limited this study to only 4 and 6 symbols. If it were deemed necessary, however, every degree on the compass could have its own symbol.

Throughout the state variations, however, the realO matrix will remain unchanged. The state is an important area to look at when determining order because this is the part of the model that can't be set arbitrarily. Furthermore, the state is also the part of a HMM that can't be observed. That makes it impossible to determine an order based strictly on seeing what the output is. Therefore, we have to find a way to determine an order based on the limited observational data from the realO matrix.

The realO matrix and calculations for the order can be seen as part of Appendix E. The premise is as follows: The probability of each sequence of realO is determined by LOGP routine in previous section. The LOGP routine is based on the parameters of the model in question. If these parameters are closer to ideal than other models, higher probabilities will be determined for the sequences in real0. This is true because more ideal parameters would predict the sequences in real0 with a higher probability. If we were to take the log of these parameters, the above strategy changes slightly. The highest probability will now be closest to zero. A set of the highest probabilities, when summed, will be closer to zero than another set of probabilities summed.

In essence, a score is generated to evaluate a model. The score closest to zero on the log scale will determine the best model. Remember, this is true because the probabilities come straight from the parameters of the model. It is not unlike doing a reduced chi squared fit. We try to minimize the vertical distance between the data and line drawn. The summed vertical distance is the reduced chi square and once that is minimized, the best fit is known [20]. Our line of fit is set at the x-axis. The Markov states of the model are the only things altered to try to fit that line.

It is important to note that this isn't really a chi-squared fit. It is only like a chi-squared fit. Chi-squared has slopes and uncertainties that play a role, whereas the vertical separation is the only important part of the above procedure. Also, this model doesn't move the line to reduce the score, the model is fundamentally changed so the data will fit the line better. It is the inverse process.

Changing the states of the model, however, requires that the training data not change. If the data were to change then comparing two different models wouldn't have any significance. That is the reason real0 was kept constant for a given set of allowed observation symbols. Finally, after all the new calculations the different states can be compared based on the real0 data. The lowest score (i.e., lowest summed vertical difference) will best determine the order of the model.

CHAPTER IV

MODEL FORMULATIONS

As mentioned above, there has only been one formulation of the model discussed. We have been using 3 Markov states and 4 observation symbols to explain the algorithm in the previous chapter. The 3 states correspond to a speed state, conceal state, and ambivalent state. Basically, the Markov states represent the mindframe that the tracked object is in. The 4 observation symbols correspond to each direction of the compass. This made the methodology simpler to explain because there was a physical significance associated with each part of the model. For this application, perhaps there are more sophisticated states that we aren't considering. Also, there might be more observation symbols than we initially allowed.

Under these circumstances, the model discussed up to now would become useless. Modeling error would be large [10]. It wouldn't be an accurate estimate of what is happening. The purpose of this chapter is to try to go past what we can lend physical significance to in Markov state modeling and determine the best order of the model. Observation symbol variation will also be explained. Regardless of the change, however, the matrices should stay axiomatically correct. All rows still have to sum to unity in order to be valid when used later on.

Only the alterations will be discussed below. It would be redundant to discuss every model totally since there is much overlap between the models. The interesting parts of each model are how they differ from the fundamental model used in the methodology chapter. A list of each model can be seen as part of table 1.

MODEL NUMBER	NUMBER OF STATES	NUMBER OF OBSERVATION SYMBOL
1	3	4
2	3	6
3	4	4
4	4	6
5	5	4
6	5	6

Table 1: List of each Model Formulation used.

Six Observations

In this study, 4 and 6 observations were examined. For this application, the number of symbols is really only limited by the computer speed. The issue is the precision needed to determine where the object will be at a specific time. Theoretically, each degree of the compass can be used if needed, but realistically that doesn't make any sense.

A trend in movement will be predicted in as few observations as possible. This study will use 3 observations. Many observation symbols requires more data.

With a very complicated model, the time it takes to predict a movement might be longer than just going out and looking for it with no prior knowledge at all. With that in mind, it might not be very useful to go much more complicated than demonstrated.

The changes needed to increase the number of observation symbols can be seen as part of Appendix F. The convenient aspect of this model is that it is set up in way that takes whatever matrices given to it and does the complicated work without being altered. In other words, the initial matrices section of the methodology is the only part of the model that needs to be changed to change the order. The rest of the methodology is general enough to adopt itself to given matrices.

The same vector field is used in the 6 symbol and 4 symbol models. The major change happens in the number of directions looked in. The encompassed angle is decreased from 90 to 60 degrees. In accordance with that, fewer rows or columns of the field are examined for any one direction. Also, the weighting had to be changed slightly to account for this decrease.

Other than that, the probabilities for each of the states are calculated the same way as in the 4 symbol model. The observation symbol matrix, B, increases columns from 4 to 6 to fit the new symbol probabilities. The A and π matrices, however, remain untouched because the number of Markov states doesn't change at all. Changing the number of symbols is really not that difficult. The rest of the formulation will pick up these matrices and operate according to how the methodology stated in the previous chapter.

Four States

Changing the number of allowed Markov states is unfortunately not as trivial. As mentioned above, we have only talked about a 3 state model. Each one of these had a physical significance. Here we have 4 states. There are 4 symbols so as to not confuse the issue more than it needs to be. Now that we know the formulation is correct (see result chapter) physical significance can play less of a role in the Markov states. The changes made can be seen as part of Appendix G.

From the given vector field, the 4 directions are calculated and applied to the speed and conceal state. The speed and conceal states were kept in this formulation as a creative choice. The ambivalent state, however, goes away. Instead, we parameterize a beta probability density function and split the ambivalent state into two separate states. The parameters are also a creative choice. The parameters were designed to make the two states compliment each other. What is the physical meaning to these 2 states? There might not be any. It is not important. Meaning can be examined and determined later if this formulation is the best model.

Parameterizing these states with observational data will adjust symbol probabilities accordingly.

What about state transition probabilities and initial state probabilities? Even though there are more states than before, this estimation will not change. State transitions will be approximated using beta density functions set up to be recurrent in one iteration. The initial state probabilities will also be estimated with a beta function parameterized to be more probable for the semi-ambivalent states.

Unlike the symbol changes mentioned above, changing the states changes all the matrices. The B matrix will take on another row to accommodate the extra state. The A matrix will be 4x4 now instead of 3x3. This is so that it can show the extra state transitions. The π matrix will take on another element to show an initial probability for the extra Markov state.

Five States

Here, instead of 4 Markov states, the model is formulated with five states. The only difference comes from how we formulate the states. The beta function is used again with the same parameters but a new state has entered the picture. Here, we chose to bring the true ambivalent state back. The extra state uses a uniform distribution as the first estimate. The alteration for 5 states can be seen in Appendix H.

CHAPTER V

RESULTS

As mentioned above, the principle behind all this work has been to come up with a novel approach to determine the order of a HMM so as to predict future movements. With those predictions we can then hopefully reduce a search area when trying to destroy an object. The model has to be validated, however, before any useful information can be taken from it

Correctness of Model

Essentially, this model is for forecasting. How do we go about validating a forecasting model? A perfectly good approach is to take data from the distant past and try to predict the data that is in the recent past [9].

The simplest way to test the model is to design several simple sequences to force a certain state prediction. If the model predicts the correct state based on the sequence, it is stable. We can design trivial sequences to force a prediction to each state of the fundamental model. The trivial sequences represent the distant past, while the state being tested represents the recent past. The fundamental model was used to test the formulation. We don't need to check the rest of the models because they are just derivations on the first. Remember that all other models only change how the initial matrices are estimated.

As long as these matrices sum rows to unity and the fundamental model works, all of the rest are correct also.

Table 2 has the sequences chosen.[13] It should be obvious by looking at Figure 2 or Appendix A that the first sequence should force predictions to the conceal state. Remember that the highest parts of the field correspond to areas of more concealment. The second sequence, on the other hand, stays with the lowest parts and should force a speed state. The lowest parts of the field corresponds to fast areas such as roads and clear weather. The third sequence makes a circle and obviously forces an ambivalent state.

Forced State	Conceal	Speed	Ambivalent
Observ. 1	0	3	0
Observ. 2	0	3	1
Observ. 3	0	0	2

Table 2: Sequences to Determine Correctness of Model

Appendix I has the results for this test. By examining the A, B, and π matrices we can see where the model predicts future movements and state transitions based on highest probabilities. γ, for example, shows what state is expected at each time. The highest probability should be with forced state. Also, we can look at what is predicted for the

[13]Remember that 0 is north, 1 is east, 2 is south, and 3 is west

future. If the most probable movement predictions mimic our trivial sequence then we know the model works. Based on the results, it is safe to say that this model does force the appropriate states. Hence, the formulation is stable and correct.

The format of Appendix I is also the format that we would use when trying to attain information about any observations. The A, B, and π matrices, γ, and future predictions matrix should provide an analyst with enough information about a target to make decisions on its future movements. In addition, the three of these results when taken together should also be enough for someone to make a decision on whether or not the data is inconclusive.

Order of Model

Now that we are confident that the basic algorithm is correct, we can look at determining the proper order. Much deliberation went into finding a method that would show the best order of a model given the limited training data. As mentioned before, an obvious way is to define a truth model and generate data from that. Then, see if the model could parameterize to that truth model over time. There are several ways to test the model depending on the application [21,5].

The problem arises in the limited amount of data. A truth model could forseeably give misleading data over short time spans. This would, in turn, cause the model to parameterize poorly and make us think that the formulation is wrong when it isn't. Therefore, we need something different.

Instead of a truth model, consider an ideal model. This ideal model will never mislead over any time span. The parameters of the ideal model are set up to perfectly predict the data that was given. In other words, plotting the log probabilities of the sequences would be very close to the y=0 line for the ideal model. The summed vertical distances of each point from the x-axis would be very close to zero.

Our challenge simply becomes finding the same log probabilities for the generated models. The model with the smallest summed vertical distances is closest to the ideal model. In other words, that model will have the most appropriate order.

Appendix J has the results of both the 4 observation models and the 6 observation models. The matrix for each model contains the sorted log probabilities of the sequences for the realO matrix. These log probabilities are based on the final parameters of the corresponding model. Remember that the realO matrix will change between 4 and 6 observations to allow for the possibility of more symbols.

Each matrix is then summed to produce a "score" for that model. The model that is closest to the ideal will have a score closest to zero because the correct model would have predicted the realO matrix perfectly. The score for a set of observations can then be graphed versus the state of the model. Figure 5 shows the 4 symbol score, while Figure 6 shows the 6 symbol scores.

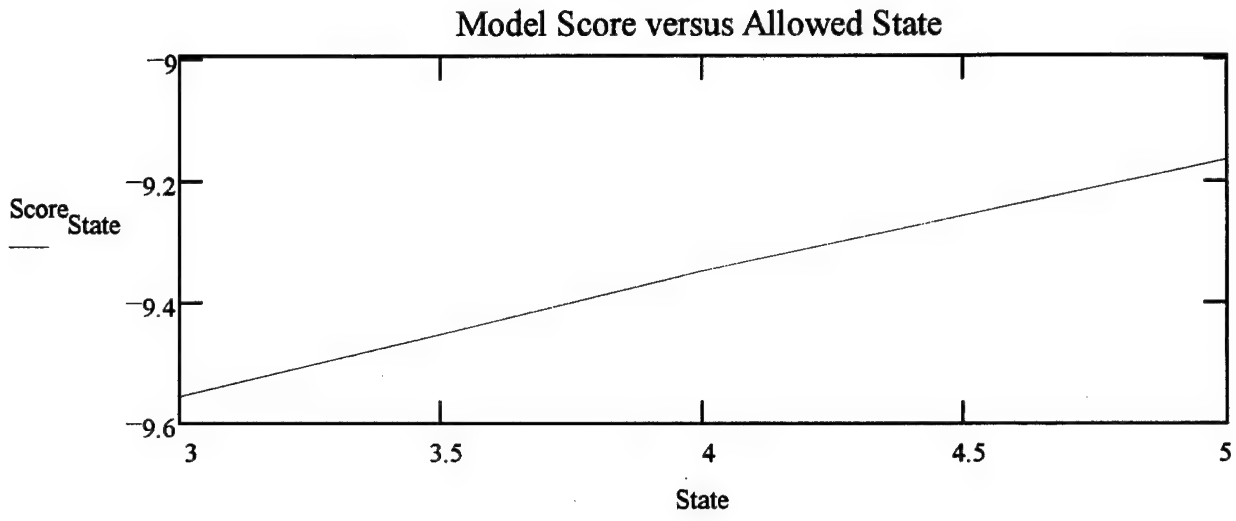


Figure 5. 4 Observational Symbol Scores versus Markov State

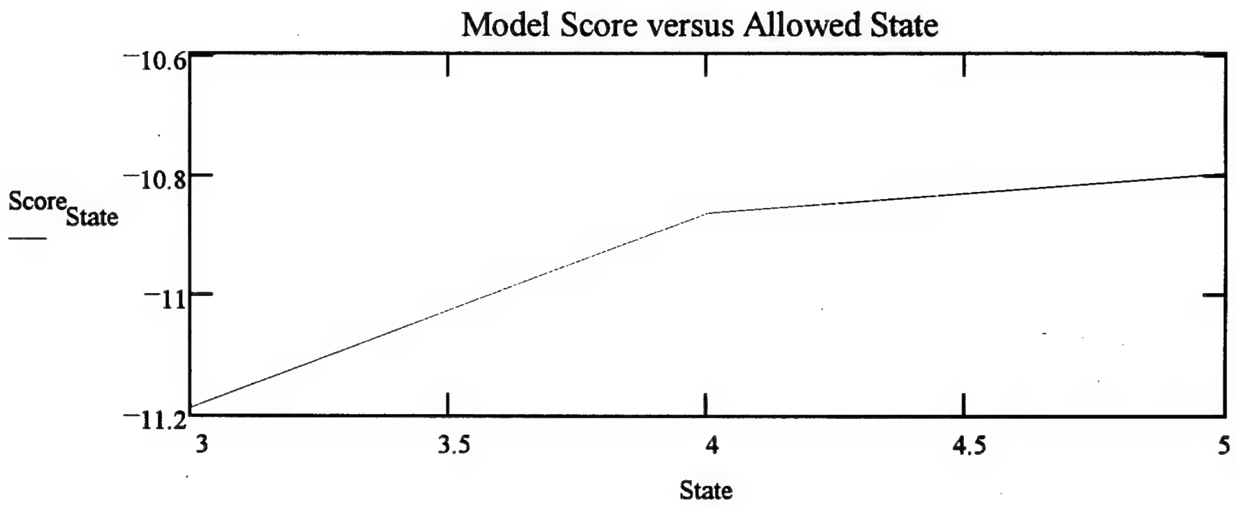


Figure 6. 6 Observational Symbol Scores versus Markov State

As the number of states increased, the score was better for the 4 symbol model. This would indicate that 5 states best describes the real0 data. As the states increased for the 6 symbol model, however, there was a plateau effect. For all intensive purposes one could make an argument that the 4 and 5 state models acted the same. Both were better than the 3 state model, however. More testing would have to be done to reach a conclusion on the order of the model in both cases.

The 4 symbol model had lower scores than the 6 symbol model. This is why we couldn't compare the two directly. As we mentioned before, more observations possible means inherently lower probabilities. Without thinking carefully about it, we might have compared the two and decided that the 4 symbol models were automatically better. This isn't necessarily the case. The nature of the two are totally different in that the more symbols we allow the more data we will need. This is a definite trade-off to consider when deciding how many symbols is appropriate for a specific application.

It is unfortunate that further testing couldn't be conducted to reach more of a conclusion. The barrier was in the run time it took for each model. The differences between simple and complex were marked. The fundamental 3 state, 4 observation model took approximately 18 minutes to run. The most complex model with 5 states and 6 observations took on the order of 8 hours to run. The number of states was the main factor in run time, while number of observation symbols was inconsequential in determining how long the model would run.

CHAPTER VI

FURTHER RESEARCH

This paper is by no means all inclusive. There are many areas to study that still need to be considered. One area is the completeness of the testing. This algorithm was not so much interested in computational elegance as it was in correctness. Investigating the long run times and trying to cut them down would allow more complex models to be run without the extremely long waits. One thing that would certainly help would be to put this algorithm into C or Fortran 90. The user friendly environment of MathCad doesn't help its efficiency. Also, restructuring the sub-routines and cutting down on redundant calculations will help with more complex models.

Secondly, the formulation of the model itself should be considered for research. The ergodic assumption we made is not true for a branch of HMM called left to right models. Once the state is transitioned from, it is never allowed back. Perhaps this type of model would better suit this application.

Thirdly, experimentally validating should be considered. Process noise and measurement noise will now play a role. This could serve to ruin the model altogether or end up being inconsequential. The algorithm won't serve any purpose if it doesn't even match what we are trying to model.

Lastly, this algorithm should be considered with other techniques as a way to predict movement. For instance, the expected Markov state could be used as a way to measure noise for a Kalman filtering system. The Kalman filter would then predict movement based on the constraints imposed by the most likely Markov state [10].

CHAPTER VII

CONCLUSION

Based on computer generated data, we used HMM to try to formulate a model to predict future movement trends. The useful outputs of the model were the A, B and π matrices, γ, the expected state at time t, and the predictions for future movements. All outputs are based on bearing only. With velocities and time intervals for specific applications positions can be inferred.

MathCad had trouble dealing with the amount of small numbers presented to it so numerical stability was an issue at a couple of different places. In spite of that, we feel that this formulation has merit and should be considered when implementing movement algorithms. The ability to model unseen Markov states can prove to be very powerful in modeling and predicting movements.

References:

[1] Atkinson on Scuds, http://ww2.pbs.org/wgbh/pages/frontline/gilf/weapons/ascud.html

[2] RadarSat Information, coorda@radar.space.gc.cs, Canadian Space Agency, 19 January 1996

[3] John T. Parish, personal interview, Sept 1997.

[4] Makhoul, John, and Richard Schwartz. "What is a hidden Markov model?" *IEEE Spectrum* Dec.1997:44-45.

[5] Fielding, Kenneth H., and Dennis W. Ruck. "Spatio-Temporal Pattern Recognition Using Hidden Markov Models." *IEEE Transactions on Aerospace and Electronic Systems* 31(1995):1292-1300.

[6] Baldi, Pierre, Yves Chavin, Tim Hunkapiller, and Marcella A. McClure. "Hidden Markov models of biological primary sequence information." *Proceedings of the National Academy of Sciences USA* 91(1994):1059.

[7] Hughey, Richard, and Anders Krogh. "Hidden Markov models for sequence analysis: extension and analysis of basic method." http://www.cse.ucsc.edu/research/c..mat/papers/hughkrogh96/cabious.html, 1996.

[8] Makhoul, John, Salim Roucos, and Herbert Gish. "Vector Quantization in Speech Coding." *Proceedings of the IEEE* 73(1985):1551.

[9] Morrison, Foster. "Model Validation." *The Art of Modeling Dynamic Systems*. New York: John Wiley and Sons, 1991. 347-357.

[10] Dr. Donald Caughlin, Personal Interview, March 1998.

[11] Hsu, Hwei P. "Random Signals and Noise." *Schwaum's outlines: Analog and Digital Communications*. New York: McGraw-Hill, 1993. 184-191.

[12] Hillier, Fredrick S., and Gerald J. Lieberman. "Markov Chains." *Introduction to Operations Research*. 6th ed. New York: McGraw-Hill, 1995. 628-660.

[13] Scheaffer, Richard L. *Introduction to Probability and its Applications*. Belmont: Duxbury Press, 1995. 6-68, 120-128, 225.

[14] Rabiner, Lawerence R. "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition." *Proceedings of the IEEE* (77) 1989:257-285.

[15] Dunning, Ted. "What makes a Hidden Markov Model hidden?" http://nora.hd.uib.no/carpora/1995-2/0047.html

[16] Levinson, S.E., L.R. Rabiner, and M.M. Sondhi. "An Introduction to the Application of the Theory of Probabilistic Functions of a Markov Process to Automatic Speech Recognition." *Proceedings of the IEEE* (62)1983:1035-1069.

[17] Forney, David G. Jr. "The Viterbi Algorithm." *Proceedings of the IEEE* (61)1972:268-273.

[18] Kiemele, Mark J., and Stephen R. Schmidt. *Basic Statistics: Tools for Continous Improvement*. 3rd ed. Colorado Springs, CO: Air Academy Press, 1993.3-28.

[19] Press, William H., Brian P. Flannery, Saul A Teukolosky, and William T. Vetterling. *Numerical Receipes in C*. Cambridge: University Press, 1986.

[20] Bevington, Philip R., and D. Keith Robinson. *Data Reduction and Error Analysis for the Physical Sciences*. 2nd ed. New York:McGraw-Hill, 1992. 96-110.

[21] Juang, B.H., and R.L. Rabiner. "A Probabilistic Distance Measure for Hidden Markov Models." *AT&T Technical Journal* (64)1985:391-402.

[22] User's Guide: MathCad 5.0 for Windows, 1994, pg 379.

APPENDIX A: ESTIMATE INITIAL PARAMETERS OF MODEL

This model is for 4 possible observation symbols and 3 possible hidden states. Throughout the text this model will be referred to as the fundamental model because it is the basic model that we will use to expand into others.

$N := 14 \qquad i := 0..N \qquad j := 0..N$

$x_i := \left(\frac{1}{N}\right)\cdot i \qquad y_j := \left(\frac{1}{N}\right)\cdot j$

$$a(x,y) := .5\cdot e^{-\left[\frac{(9\cdot x-8)^2+(9\cdot y-4)^2}{4}\right]}$$

These functions describe each of the hills in the z-function below.

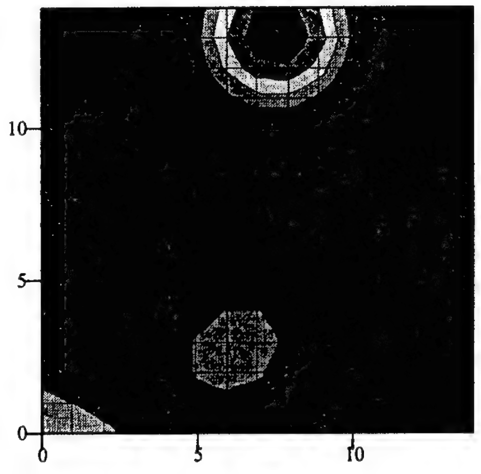
$$b(x,y) := .85\cdot e^{-\left[\frac{(9\cdot x-5)^2+(9\cdot y-8.5)^2}{4}\right]}$$

General form of these equations was taken out of the MathCad 5.0 reference manual.

$$f(x,y) := .34\cdot e^{-\left[\frac{(9\cdot x-4.5)^2+(9\cdot y-2)^2}{4}\right]} + .75\cdot e^{\left[\frac{(9\cdot x+1)^2}{-49} - \left(\frac{9\cdot y+1}{10}\right)\right]} + b(x,y) + a(x,y)$$

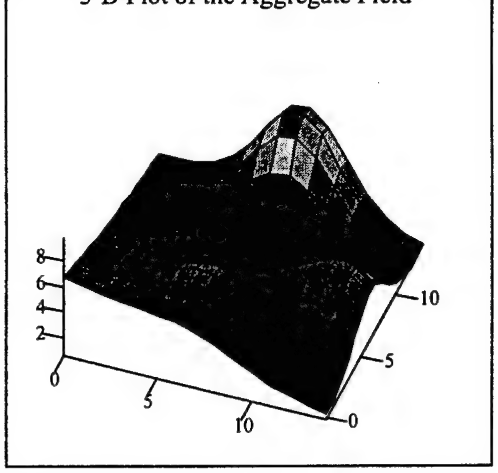
$$M_{i,j} := 10\cdot f(x_i, y_j)$$

Contour Plot of Aggregate Vector Field



M

3-D Plot of the Aggregate Field



M

x dwn y across

In the above contour plot, height will be proportional to high probabilities in the conceal state, where as the inverse will be proportional to high probabilties in speed state.

$M_{i,j} := M_{i,j} - M_{7,7}$

In order to start at gnd zero, we need to difference all the elements with the middle one where we start the vehicle at.

east is up,
west is down.
north is right,
south is left.

M =

3.245	2.836	2.453	2.092	1.749	1.424	1.118	0.832
3.04	2.662	2.308	1.966	1.63	1.302	0.99	0.701
2.782	2.47	2.178	1.874	1.541	1.189	0.845	0.536
2.524	2.34	2.17	1.935	1.593	1.17	0.74	0.371
2.312	2.34	2.369	2.242	1.871	1.313	0.723	0.243
2.138	2.437	2.723	2.734	2.323	1.585	0.782	0.156
1.91	2.458	2.985	3.131	2.698	1.813	0.829	0.08
1.503	2.179	2.836	3.082	2.693	1.801	0.781	0
4.282	4.92	5.565	5.876	5.647	4.964	4.116	3.411
3.506	3.993	4.544	4.935	5.005	4.743	4.246	3.674
2.744	3.071	3.544	4.077	4.558	4.84	4.769	4.311
2.109	2.335	2.791	3.51	4.409	5.209	5.519	5.121
1.625	1.806	2.27	3.12	4.292	5.422	5.968	5.599
1.256	1.407	1.84	2.676	3.863	5.032	5.628	5.301
0.961	1.073	1.412	2.081	3.039	3.991	4.484	4.228

These numbers correspond to the heights of the above z-function.

This is the matrix that will be used in order to find the highest area in each direction. The routines below will find the weighted sum across the respective look angles. The routines are unique for each direction.

```
west :=
    $x \leftarrow 0$
    while $x \le 14$
        $max_x \leftarrow 0$
        $x \leftarrow x + 1$
    $a \leftarrow 4$
    $b \leftarrow 9$
    while $a \ge 0$
        $max_b \leftarrow M_{a,b}$ if $|M_{a,b}| > |max_b|$
        $a \leftarrow a - 1$
    $max_b \leftarrow max_b \cdot .1$
    $a \leftarrow 5$
    $b \leftarrow 8$
    while $a \ge 0$
        $max_b \leftarrow M_{a,b}$ if $|M_{a,b}| > |max_b|$
        $a \leftarrow a - 1$
    $max_b \leftarrow max_b \cdot .2$
    $a \leftarrow 6$
    $b \leftarrow 7$
    while $a \ge 0$
        $max_b \leftarrow M_{a,b}$ if $|M_{a,b}| > |max_b|$
        $a \leftarrow a - 1$
    $max_b \leftarrow max_b \cdot .4$
    $a \leftarrow 5$
    $b \leftarrow 6$
    while $a \ge 0$
        $max_b \leftarrow M_{a,b}$ if $|M_{a,b}| > |max_b|$
        $a \leftarrow a - 1$
    $max_b \leftarrow max_b \cdot .2$
    $a \leftarrow 4$
    $b \leftarrow 5$
    while $a \ge 0$
        $max_b \leftarrow M_{a,b}$ if $|M_{a,b}| > |max_b|$
        $a \leftarrow a - 1$
    $max_b \leftarrow max_b \cdot .1$
    $max \leftarrow \sum_{j=0}^{14} max_j$
    $max \leftarrow .5$ if $max < 0$
```

```
east :=
    $x \leftarrow 0$
    while $x \le 14$
        $max_x \leftarrow 0$
        $x \leftarrow x + 1$
    $a \leftarrow 10$
    $b \leftarrow 9$
    while $a \le 14$
        $max_b \leftarrow M_{a,b}$ if $|M_{a,b}| > |max_b|$
        $a \leftarrow a + 1$
    $max_b \leftarrow max_b \cdot .1$
    $a \leftarrow 9$
    $b \leftarrow 8$
    while $a \le 14$
        $max_b \leftarrow M_{a,b}$ if $|M_{a,b}| > |max_b|$
        $a \leftarrow a + 1$
    $max_b \leftarrow max_b \cdot .2$
    $a \leftarrow 8$
    $b \leftarrow 7$
    while $a \le 14$
        $max_b \leftarrow M_{a,b}$ if $|M_{a,b}| > |max_b|$
        $a \leftarrow a + 1$
    $max_b \leftarrow max_b \cdot .4$
    $a \leftarrow 9$
    $b \leftarrow 6$
    while $a \le 14$
        $max_b \leftarrow M_{a,b}$ if $|M_{a,b}| > |max_b|$
        $a \leftarrow a + 1$
    $max_b \leftarrow max_b \cdot .2$
    $a \leftarrow 10$
    $b \leftarrow 5$
    while $a \le 14$
        $max_b \leftarrow M_{a,b}$ if $|M_{a,b}| > |max_b|$
        $a \leftarrow a + 1$
    $max_b \leftarrow max_b \cdot .1$
    $max \leftarrow \sum_{j=0}^{14} max_j$
    $max \leftarrow .5$ if $max < 0$
```

```
south := $x \leftarrow 0$
         while $x \le 14$
             $max_x \leftarrow 0$
             $x \leftarrow x + 1$
         $b \leftarrow 4$
         $a \leftarrow 9$
         while $b \ge 0$
             $max_a \leftarrow M_{a,b}$ if $|M_{a,b}| > |max_a|$
             $b \leftarrow b - 1$
         $max_a \leftarrow max_a \cdot .1$
         $b \leftarrow 5$
         $a \leftarrow 8$
         while $b \ge 0$
             $max_a \leftarrow M_{a,b}$ if $|M_{a,b}| > |max_a|$
             $b \leftarrow b - 1$
         $max_a \leftarrow max_a \cdot .2$
         $b \leftarrow 6$
         $a \leftarrow 7$
         while $b \ge 0$
             $max_a \leftarrow M_{a,b}$ if $|M_{a,b}| > |max_a|$
             $b \leftarrow b - 1$
         $max_a \leftarrow max_a \cdot .4$
         $b \leftarrow 5$
         $a \leftarrow 6$
         while $b \ge 0$
             $max_a \leftarrow M_{a,b}$ if $|M_{a,b}| > |max_a|$
             $b \leftarrow b - 1$
         $max_a \leftarrow max_a \cdot .2$
         $b \leftarrow 4$
         $a \leftarrow 5$
         while $b \ge 0$
             $max_a \leftarrow M_{a,b}$ if $|M_{a,b}| > |max_a|$
             $b \leftarrow b - 1$
         $max_a \leftarrow max_a \cdot .1$
         $max \leftarrow \sum_{j=0}^{14} max_j$
         $max \leftarrow .5$ if $max < 0$
```

```
north := $x \leftarrow 0$
         while $x \le 14$
             $max_x \leftarrow 0$
             $x \leftarrow x + 1$
         $b \leftarrow 10$
         $a \leftarrow 9$
         while $b \le 14$
             $max_a \leftarrow M_{a,b}$ if $|M_{a,b}| > |max_a|$
             $b \leftarrow b + 1$
         $max_a \leftarrow max_a \cdot .1$
         $b \leftarrow 9$
         $a \leftarrow 8$
         while $b \le 14$
             $max_a \leftarrow M_{a,b}$ if $|M_{a,b}| > |max_a|$
             $b \leftarrow b + 1$
         $max_a \leftarrow max_a\ .2$
         $b \leftarrow 8$
         $a \leftarrow 7$
         while $b \le 14$
             $max_a \leftarrow M_{a,b}$ if $|M_{a,b}| > |max_a|$
             $b \leftarrow b + 1$
         $max_a \leftarrow max_a \cdot .4$
         $b \leftarrow 9$
         $a \leftarrow 6$
         while $b \le 14$
             $max_a \leftarrow M_{a,b}$ if $|M_{a,b}| > |max_a|$
             $b \leftarrow b + 1$
         $max_a \leftarrow max_a \cdot .2$
         $b \leftarrow 10$
         $a \leftarrow 5$
         while $b \le 14$
             $max_a \leftarrow M_{a,b}$ if $|M_{a,b}| > |max_a|$
             $b \leftarrow b + 1$
         $max_a \leftarrow max_a \cdot .1$
         $max \leftarrow \sum_{j=0}^{14} max_j$
         $max \leftarrow .5$ if $max < 0$
```

$west = 0.844$ $east = 5.205$

$south = 3.808$ $north = 7.757$

$totconceal := north + east + south + west$

totconceal will be used as the normalizer once all the respective scores are added up.

$totconceal = 17.614$

for conceal state:

$$\frac{north}{totconceal} = 0.44$$

$$\frac{west}{totconceal} = 0.048 \quad \wedge \quad \frac{east}{totconceal} = 0.296$$

$$\frac{south}{totconceal} = 0.216$$

for speed state:

$totspd := north^{-1} + east^{-1} + south^{-1} + west^{-1}$

$totspd = 1.769$

$$\frac{north^{-1}}{totspd} = 0.073$$

$$\frac{west^{-1}}{totspd} = 0.67 \quad \wedge \quad \frac{east^{-1}}{totspd} = 0.109$$

$$\frac{south^{-1}}{totspd} = 0.148$$

so, we can see how the "scores" for each direction add up to form probabilities for the respective Markov state.

let 0 be the conceal state and 2 be the speed state. North:0 east:1 south:2 west:3

$$B := \begin{bmatrix} \frac{north}{totconceal} & \frac{east}{totconceal} & \frac{south}{totconceal} & \frac{west}{totconceal} \\ .25 & .25 & .25 & .25 \\ \frac{north^{-1}}{totspd} & \frac{east^{-1}}{totspd} & \frac{south^{-1}}{totspd} & \frac{west^{-1}}{totspd} \end{bmatrix}$$

$$B = \begin{bmatrix} 0.44 & 0.296 & 0.216 & 0.048 \\ 0.25 & 0.25 & 0.25 & 0.25 \\ 0.073 & 0.109 & 0.148 & 0.67 \end{bmatrix}$$

This will be the initial B matrix we use to train the model. Notice the uniform distribution in the ambiguous state.

$w := 0..2$ Define range variable to make sure the rows of these estimates sum to unity.

$$\sum_{j=0}^{3} B_{w,j}$$

$$\begin{array}{|c|} \hline 1 \\ \hline 1 \\ \hline 1 \\ \hline \end{array}$$

summing the B matrix across the rows yields one. This lets us know that the matrix is valid to use.

$M := 4$ define the number of observations possible.

$N := 3$ This is the number of Markov States allowed.

$i := 0..N-1$

$j := 0..N-1$

$\alpha := 3$ Define the beta distribution and the parameters we will use.

$\beta := 3$

$$f(x,\alpha,\beta) := \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\cdot\Gamma(\beta)}\cdot x^{\alpha-1}\cdot(1-x)^{\beta-1}$$

Now we have to find the state transition matrix (A) first guess. Beta distribution will be used because it normalizes to one and is easily parameterized. We only need three integrations because there are only three states to transition to.

$$\int_{0}^{\frac{1}{N}} f(x,\alpha,\beta)\,dx = 0.21 \qquad \int_{\frac{1}{N}}^{2\cdot\left(\frac{1}{N}\right)} f(x,\alpha,\beta)\,dx = 0.58$$

$$\int_{2\cdot\left(\frac{1}{N}\right)}^{3\cdot\left(\frac{1}{N}\right)} f(x,\alpha,\beta)\,dx = 0.21$$

Here we have an example to make sure we are doing things properly.

This is a more generalized form of the integrations above. Another sample calculation tells us that we are still okay.

$$\xi(v,\alpha,\beta) := \int_{(v-1)\cdot\frac{1}{N}}^{v\cdot\left(\frac{1}{N}\right)} f(x,\alpha,\beta)\,dx \qquad \xi(3,\alpha,\beta) = 0.21$$

$$A := \begin{bmatrix} \xi(1,2,4) & \xi(2,2,4) & \xi(3,2,4) \\ \xi(1,3,3) & \xi(2,3,3) & \xi(3,3,3) \\ \xi(1,2,3) & \xi(2,2,3) & \xi(3,2,3) \end{bmatrix} \qquad A = \begin{bmatrix} 0.539 & 0.416 & 0.045 \\ 0.21 & 0.58 & 0.21 \\ 0.407 & 0.481 & 0.111 \end{bmatrix}$$

$$A(a,b,c,d,e,f,N) := \begin{bmatrix} \xi(N-2,a,b) & \xi(N-1,a,b) & \xi(N,a,b) \\ \xi(N-2,c,d) & \xi(N-1,c,d) & \xi(N,c,d) \\ \xi(N-2,e,f) & \xi(N-1,e,f) & \xi(N,e,f) \end{bmatrix}$$

All the above manipulation was either putting it into an easy form MathCad will agree with or checking things out with myself that they actually work.

$$A(2,4,3,3,4,2,3) = \begin{bmatrix} 0.539 & 0.416 & 0.045 \\ 0.21 & 0.58 & 0.21 \\ 0.045 & 0.416 & 0.539 \end{bmatrix}$$

This is the form of the A matrix that we will now work with as an initial matrix

$$A := A(2,4,3,3,4,2,3)$$

$$A = \begin{bmatrix} 0.539 & 0.416 & 0.045 \\ 0.21 & 0.58 & 0.21 \\ 0.045 & 0.416 & 0.539 \end{bmatrix}$$

notice we set up the initial matrix so the chance of reentering the same state is the highest.

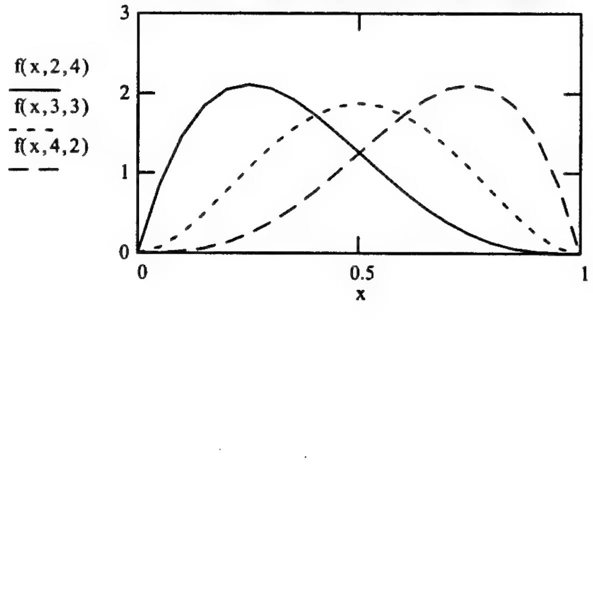
$$\sum_{j=0}^{2} A_{w,j}$$

$$\begin{array}{|c|} \hline 1 \\ \hline 1 \\ \hline 1 \\ \hline \end{array}$$

summing the A matrix across the rows yields one. This lets us know that the matrix is valid to use.

$x := 0, .05 .. 1$ define a range variable to plot the beta functions below.

Plot of Beta Functions for each State



here we have the initial beta functions with parameters for the state transition matrix. These were parameterized this way because they estimate recurrent states being more probable than non-recurrent states.

Now the initial state probability matrix (π) has to be determined. It is the same as the state transition matrix but, it only has one row.

$\alpha := 2$

$\beta := 2$

$$f(x,\alpha,\beta) := \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\cdot\Gamma(\beta)}\cdot x^{\alpha-1}\cdot(1-x)^{\beta-1}$$

Show the beta function and parameters again.

$$\int_{0}^{\frac{1}{N}} f(x,\alpha,\beta)\,dx = 0.259$$

$$\int_{\frac{1}{N}}^{2\cdot\left(\frac{1}{N}\right)} f(x,\alpha,\beta)\,dx = 0.481$$

$$\int_{2\cdot\left(\frac{1}{N}\right)}^{3\cdot\left(\frac{1}{N}\right)} f(x,\alpha,\beta)\,dx = 0.259$$

Again, here we have some sample calculations to make sure the equations are set up properly and the general form below it.

$$\xi(v,\alpha,\beta) := \int_{(v-1)\cdot\frac{1}{N}}^{v\cdot\left(\frac{1}{N}\right)} f(x,\alpha,\beta)\,dx \qquad \xi(3,\alpha,\beta) = 0.259$$

$$\pi := (\xi(1,2,2)\ \ \xi(2,2,2)\ \ \xi(3,2,2))$$

$$\pi = [\,0.259\ \ 0.481\ \ 0.259\,]$$

$$\pi(a,b,N) := (\xi(N-2,a,b)\ \ \xi(N-1,a,b)\ \ \xi(N,a,b))$$

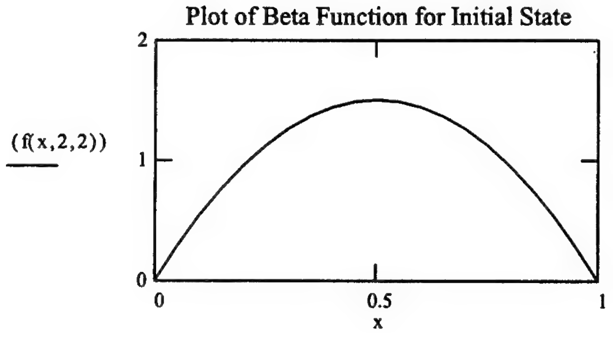
$\pi(2,2,3) = [\,0.259\ \ 0.481\ \ 0.259\,]$ Here we set the most probable state to start in as the ambiguous state.

$\pi := \pi(2,2,3)$ Again, the above manipulations are just to get the notation in a form that is easy to use later on.

$$\sum_{j=0}^{2} \pi_{0,j} = 1$$

summing the π matrix across the row yields one. This lets us know that the matrix is valid to use.

$x := 0, .05 .. 1$



here we have the beta function with parameter for the initial state probability matrix.

APPENDIX B: Generate Data

The T tells us how long each observation set is, K tells us how many observation sets to look at, and Real tells us how many sets of real observations we have.

$T := 3$ $K := 2$ $Real := 5$

$t := 0..T$ $k := 0..K$ $real := 0..Real$ $time := 0..2$

The function below will give us random numbers but they are all based on the beta distribution. The parameters are a creative choice.

The time range variable is used below in finding the order of the model.

$r := 2$ $s := 2$

$obs^{\langle k \rangle} := 3.0 \cdot rbeta(T + 1, r, s)$

We are only interested in integers. That is why must round the number generator to make the data useful.

$round(x) := if(x - floor(x) < .5, floor(x), ceil(x))$

The real observation data will be used later on to determine the order of the model. The obs data taken above is the data used to train the model.

$O_{t,k} := round\left(obs_{t,k}\right)$

$realobs^{\langle real \rangle} := 3.5 \cdot rbeta(time + 1, r, s)$

$realO_{time,real} := round\left(realobs_{time,real}\right)$

$$O := \begin{bmatrix} 2 & 1 & 2 \\ 1 & 2 & 1 \\ 1 & 1 & 3 \\ 1 & 1 & 2 \end{bmatrix}$$

$$realO := \begin{bmatrix} 1 & 2 & 2 & 2 & 2 & 1 \\ 1 & 0 & 3 & 1 & 1 & 3 \\ 2 & 1 & 2 & 3 & 2 & 2 \end{bmatrix}$$

realO has to be three elements long. This is true because it is what we use below to determine the order. That part of the algorithm uses 3 elements exclusively to find movement trends.

APPENDIX C: Forward and Backward Variable

```
FwdProb(col) := | i ← 0
                | j ← 0
                | set ← col
                | while i ≤ N − 1
                |     | $\alpha_{i,0} \leftarrow \pi_{0,i} \cdot B_{i,\left(O^{\langle set \rangle}\right)_0}$
                |     | i ← i + 1
                | while j ≤ N − 1
                |     | t ← 0
                |     | while t ≤ T − 1
                |     |     | $sum \leftarrow \sum_{s=0}^{N-1} \alpha_{s,t} \cdot A_{s,j}$
                |     |     | $\alpha_{j,t+1} \leftarrow sum \cdot B_{j,\left(O^{\langle set \rangle}\right)_{t+1}}$
                |     |     | t ← t + 1
                |     | j ← j + 1
                | α
```

Let's look at just the last column of data from Obs matrix to show this variable. As time progresses across the rows we can see how small the probabilities get for the observed data.

$$\mathrm{FwdProb}(0) = \begin{bmatrix} 0.056 & 0.017 & 2.694 \cdot 10^{-3} & 4.292 \cdot 10^{-4} \\ 0.12 & 0.027 & 5.715 \cdot 10^{-3} & 1.109 \cdot 10^{-3} \\ 0.038 & 5.273 \cdot 10^{-3} & 1.014 \cdot 10^{-3} & 2.029 \cdot 10^{-4} \end{bmatrix}$$

```
BackProb(col) := | i ← 0
                 | set ← col
                 | while i ≤ N − 1
                 |     | $\beta_{i,T} \leftarrow 1$
                 |     | i ← i + 1
                 | t ← T − 1
                 | while t ≥ 0
                 |     | i ← 0
                 |     | while i ≤ N − 1
                 |     |     | $\beta_{i,t} \leftarrow \sum_{j=0}^{N-1} A_{i,j} \cdot B_{j,\left(O^{\langle set \rangle}\right)_{t+1}} \cdot \beta_{j,t+1}$
                 |     |     | i ← i + 1
                 |     | t ← t − 1
                 | β
```

Again, let's look only at the first column of data. The sample matrix shows us starting at one and working back.

$$\text{BackProb}(0) = \begin{bmatrix} 0.017 & 0.067 & 0.268 & 1 \\ 0.013 & 0.054 & 0.23 & 1 \\ 8.724 \cdot 10^{-3} & 0.038 & 0.176 & 1 \end{bmatrix}$$

As time increases we will have problems with underflow. This can be fixed by using scaling factors below. Both methods should produce the same results.

$set := 0$ specify a column so we can compare apples to apples below.

$$scale_t := \frac{1}{\sum_{i=0}^{N-1} FwdProb(set)_{i,t}}$$

these two functions are the same, they just are set up differently.

```
c(col) := | set ← col
          | t ← 0
          | while t ≤ T
          |     | $c_t \leftarrow \frac{1}{\sum_{i=0}^{N-1} FwdProb(set)_{i,t}}$
          |     | t ← t + 1
          | c
```

$$scale = \begin{bmatrix} 4.653 \\ 20.215 \\ 106.122 \\ 574.347 \end{bmatrix} \qquad c(0) = \begin{bmatrix} 4.653 \\ 20.215 \\ 106.122 \\ 574.347 \end{bmatrix}$$

Two methods were used to check work. Both are the same so we can use either one to fix underflow problems later on. Now, let's add them to the Forward and Backward variables.

```
FwdProbhat(col) := | set ← col
                   | i ← 0
                   | while i ≤ N − 1
                   |     | t ← 0
                   |     | while t ≤ T
                   |     |     | $FwdProbhat_{i,t} \leftarrow c(set)_t \cdot FwdProb(set)_{i,t}$
                   |     |     | t ← t + 1
                   |     | i ← i + 1
                   | FwdProbhat
```

Hat notation indicates that the scale has been added in.

$$FwdProbhat(0) = \begin{bmatrix} 0.261 & 0.342 & 0.286 & 0.247 \\ 0.56 & 0.552 & 0.607 & 0.637 \\ 0.179 & 0.107 & 0.108 & 0.117 \end{bmatrix}$$

scale term restores the magnitude of the sum downward to 1.

```
BackProbhat(col) := set ← col
                    i ← 0
                    while i ≤ N − 1
                        t ← 0
                        while t ≤ T
                            $\text{BackProbhat}_{i,t} \leftarrow c(\text{set})_t \cdot \text{BackProb}(\text{set})_{i,t}$
                            t ← t + 1
                        i ← i + 1
                    BackProbhat
```

Again we used the same column of data. These numbers go well over one because the scale is based on the forward variable. For implementation purposes, it is better to use these large numbers than the very small numbers that might cause underflow.

$$\text{BackProbhat}(0) = \begin{bmatrix} 0.077 & 1.364 & 28.456 & 574.347 \\ 0.06 & 1.091 & 24.395 & 574.347 \\ 0.041 & 0.764 & 18.66 & 574.347 \end{bmatrix}$$

Now, we should look at what Markov state is most likely at any particular time.

$u := 0$ u is the column of data in the Obs matrix we specify for example purposes.

$$\gamma_{i,t} := \frac{\left(\mathrm{FwdProb}(u)_{i,t} \cdot \mathrm{BackProb}(u)_{i,t}\right)}{\left[\sum_{i=0}^{N-1} \left(\mathrm{FwdProb}(u)_{i,t} \cdot \mathrm{BackProb}(u)_{i,t}\right)\right]}$$

here we can see the most likely state at each time. Slightly different optimization criteria than the Viterbi but I like it.

$$\sum_{i=0}^{N-1} \gamma_{i,t}$$

$$\begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

$$\gamma = \begin{bmatrix} 0.33 & 0.406 & 0.326 & 0.247 \\ 0.551 & 0.524 & 0.593 & 0.637 \\ 0.119 & 0.071 & 0.081 & 0.117 \end{bmatrix}$$

only one of the columns of observations was picked here for example purposes, but they all add up to one so that gives us a good feeling that all is right in the way the multiple sequences is set up.

As an additional check, look at the γ variable again with the scale added to it. It should output the same matrix.

$$\gamma_{i,t} := \frac{\left(\mathrm{FwdProbhat}(u)_{i,t} \cdot \mathrm{BackProbhat}(u)_{i,t}\right)}{\left[\sum_{i=0}^{N-1} \left(\mathrm{FwdProbhat}(u)_{i,t} \cdot \mathrm{BackProbhat}(u)_{i,t}\right)\right]}$$

so, now with the scaling constants added we see that they cancel out each other and give us the exact answer above. It works!

$$\sum_{i=0}^{N-1} \gamma_{i,t}$$

$$\begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

$$\gamma = \begin{bmatrix} 0.33 & 0.406 & 0.326 & 0.247 \\ 0.551 & 0.524 & 0.593 & 0.637 \\ 0.119 & 0.071 & 0.081 & 0.117 \end{bmatrix}$$

at 0:

$$\Xi atZero_{i,j} := \frac{BackProbhat(u)_{j,1} \cdot A_{i,j} \cdot B_{j,\left(O^{<u>}\right)_1} \cdot FwdProbhat(u)_{i,0}}{\sum_{i=0}^{N-1} \sum_{j=0}^{N-1} FwdProbhat(u)_{i,0} \cdot BackProbhat(u)_{j,1} \cdot A_{i,j} \cdot B_{j,\left(O^{<u>}\right)_1}}$$

$$\sum_{j=0}^{N-1} \Xi atZero_{i,j} = \begin{array}{|c|}\hline 0.33 \\ \hline 0.551 \\ \hline 0.119 \\ \hline\end{array} \qquad \gamma_{i,0} = \begin{array}{|c|}\hline 0.33 \\ \hline 0.551 \\ \hline 0.119 \\ \hline\end{array}$$

$$C_t := \prod_{\tau=0}^{t} scale_\tau \qquad D_t := \prod_{\tau=t}^{T} scale_\tau$$

We will use a different way than matrix algorithm to check the output.

$$TestA_{i,j} := \frac{\sum_{t=0}^{T-1} C_t \cdot FwdProb(0)_{i,t} \cdot A_{i,j} \cdot B_{j,\left(O^{<0>}\right)_{t+1}} \cdot BackProb(0)_{j,t+1} \cdot D_{t+1}}{\sum_{t=0}^{T-1} \sum_{j=0}^{N-1} FwdProb(0)_{i,t} \cdot A_{i,j} \cdot B_{j,\left(O^{<0>}\right)_{t+1}} \cdot BackProb(0)_{j,t+1} \cdot C_t \cdot D_{t+1}}$$

$$TestA = \begin{bmatrix} 0.629 & 0.358 & 0.014 \\ 0.303 & 0.618 & 0.079 \\ 0.094 & 0.627 & 0.279 \end{bmatrix}$$

after the matrix routine calculates the two A matrices should be the same. They are in this case to about 3 one-thousandths. That is close enough for Government work. If we were to shorten the time interval by one the elements would be totally equal. MathCad begins to fail us numerically with bigger time step.

This is part of the parameter re-estimation. A new variable has to be defined to make sure we don't go out of bounds in the re-estimation algorithm.

$\tau := 0..T-1$

$K := 0$ Here we use only the first column because we want to check by doing it a different way with the TestA above.

```
Matrix := | u ← 0
          | while u ≤ K
          |   | τ ← 0
          |   | i ← 0
          |   | j ← 0
          |   | x ← 0
          |   | y ← 0
          |   | while x ≤ N − 1
          |   |   | while y ≤ N − 1
          |   |   |   | $sumZeta_{x,y} \leftarrow 0$
          |   |   |   | y ← y + 1
          |   |   | y ← 0
          |   |   | x ← x + 1
          |   | while τ ≤ T − 1
          |   |   | while i ≤ N − 1
          |   |   |   | while j ≤ N − 1
          |   |   |   |   | $\Xi_{i,j} \leftarrow \frac{BackProbhat(u)_{j,\tau+1} \cdot A_{i,j} \cdot B_{j,\left(O^{<u>}\right)_{\tau+1}} \cdot FwdProbhat(u)_{i,\tau}}{\sum_{a=0}^{N-1} \sum_{b=0}^{N-1} FwdProbhat(u)_{a,\tau} \cdot BackProbhat(u)_{b,\tau+1} \cdot A_{a,b} \cdot B_{b,\left(O^{<u>}\right)_{\tau+}}}$
          |   |   |   |   | $sumZeta_{i,j} \leftarrow sumZeta_{i,j} + \Xi_{i,j}$
          |   |   |   |   | j ← j + 1
          |   |   |   | i ← i + 1
          |   |   |   | j ← 0
          |   |   | τ ← τ + 1
          |   |   | i ← 0
          |   | t ← 0
          |   | while t ≤ T
          |   |   | while i ≤ N − 1
          |   |   |   | $\gamma_{i,t} \leftarrow \frac{\left(FwdProbhat(u)_{i,t} \cdot BackProbhat(u)_{i,t}\right)}{\left[\sum_{i=0}^{N-1} \left(FwdProbhat(u)_{i,t} \cdot BackProbhat(u)_{i,t}\right)\right]}$
          |   |   |   | i ← i + 1
          |   |   | t ← t + 1
          |   |   | i ← 0
          |   | while i ≤ N − 1
          |   |   | $sumGamma_i \leftarrow \sum^{T-1} \gamma_{i,t}$
```

```
        sumGamma_i $\sum_{t=0} \gamma_{i,t}$
        i ← i + 1
    j ← 0
    i ← 0
    x ← 0
    y ← 0
    while x ≤ N − 1
        while y ≤ N − 1
            $Abar_{x,y} \leftarrow 0$
            y ← y + 1
        y ← 0
        x ← x + 1
    while j ≤ N − 1
        while i ≤ N − 1
            $Abar_{i,j} \leftarrow \frac{sumZeta_{i,j}}{sumGamma_i}$
            i ← i + 1
        j ← j + 1
        i ← 0
    i ← 0
    while i ≤ N − 1
        $\pi bar_{0,i} \leftarrow \gamma_{i,0}$
        i ← i + 1
    i ← 0
    while i ≤ N − 1
        $newsumGamma_i \leftarrow \sum_{t=0}^{T} \gamma_{i,t}$
        i ← i + 1
    t ← 0
    i ← 0
    x ← 0
    y ← 0
    while x ≤ N − 1
        while y ≤ M − 1
            $Y_{x,y} \leftarrow 0$
            y ← y + 1
        y ← 0
        x ← x + 1
    while i ≤ N − 1
        while t ≤ T
            $Y_{i,(O^{<u>})_t} \leftarrow Y_{i,(O^{<u>})_t} + \gamma_{i,t}$
            i, 0   . t   i, 0   . t
            $t \leftarrow t+1$
        $i \leftarrow i+1$
        $t \leftarrow 0$
    $i \leftarrow 0$
    $j \leftarrow 0$
    $\varepsilon \leftarrow .02$
    while $i \le N-1$
        while $j \le M-1$
            $Bbar_{i,j} \leftarrow \frac{Y_{i,j}}{newsumGamma_i}$
            $Bbar_{i,j} \leftarrow \varepsilon$  if $Bbar_{i,j} < \varepsilon$
            $j \leftarrow j+1$
        $normalizer \leftarrow \sum_{L=0}^{M-1} Bbar_{i,L}$
        $j \leftarrow 0$
        while $j \le M-1$
            $Bbar_{i,j} \leftarrow \frac{Bbar_{i,j}}{normalizer}$
            $j \leftarrow j+1$
        $i \leftarrow i+1$
        $j \leftarrow 0$
    $B \leftarrow Bbar$
    $\pi \leftarrow \pi bar$
    $A \leftarrow Abar$
    $u \leftarrow u+1$
$x \leftarrow 0$
$y \leftarrow 0$
while $x \le N-1$
    while $y \le 2 \cdot N + M - 1$
        $out_{x,y} \leftarrow 0$
        $y \leftarrow y+1$
    $y \leftarrow 0$
    $x \leftarrow x+1$
$x \leftarrow 0$
$y \leftarrow 0$
while $x \le N-1$
    while $y \le N-1$
        $out_{x,y} \leftarrow A_{x,y}$
        $y \leftarrow y+1$
    $y \leftarrow 0$
    $x \leftarrow x+1$
```

```
$x \leftarrow 0$
$y \leftarrow N$
while $x \le N - 1$
    while $y \le N + M - 1$
        $out_{x,y} \leftarrow B_{x,(y-N)}$
        $y \leftarrow y + 1$
    $y \leftarrow N$
    $x \leftarrow x + 1$
$x \leftarrow 0$
$y \leftarrow N + M$
while $y \le N + M + N - 1$
    $out_{x,y} \leftarrow \pi_{0,(y-(N+M))}$
    $y \leftarrow y + 1$
out
```

Here we output all the matrices as an aggregate because this is the way MathCad programming structures work. The programming structures below it simply extract the proper matrix form the aggregate.

$$\text{Matrix} = \begin{array}{|lllllllll|}\hline 0.626 & 0.36 & 0.014 & 0.019 & 0.719 & 0.242 & 0.019 & 0.33 & 0.551 \\ \hline 0.3 & 0.619 & 0.081 & 0.019 & 0.732 & 0.23 & 0.019 & 0 & 0 \\ \hline 0.093 & 0.625 & 0.282 & 0.019 & 0.665 & 0.296 & 0.019 & 0 & 0 \\ \hline \end{array}$$

```
A := | i ← 0
     | while i ≤ N − 1
     |     | $A^{\langle i \rangle} \leftarrow \text{Matrix}^{\langle i \rangle}$
     |     | i ← i + 1
     | A
```

```
B := | i ← N
     | while i ≤ N + M − 1
     |     | $B^{\langle i-N \rangle} \leftarrow \text{Matrix}^{\langle i \rangle}$
     |     | i ← i + 1
     | B
```

$$A = \begin{bmatrix} 0.626 & 0.36 & 0.014 \\ 0.3 & 0.619 & 0.081 \\ 0.093 & 0.625 & 0.282 \end{bmatrix} \qquad B = \begin{bmatrix} 0.019 & 0.719 & 0.242 & 0.019 \\ 0.019 & 0.732 & 0.23 & 0.019 \\ 0.019 & 0.665 & 0.296 & 0.019 \end{bmatrix}$$

$$\sum_{j=0}^{2} A_{i,j} \quad \begin{array}{|c|}\hline 1 \\ \hline 1 \\ \hline 1 \\ \hline \end{array}$$

Rows sum to one here!

$$\sum_{j=0}^{3} B_{i,j} \quad \begin{array}{|c|}\hline 1 \\ \hline 1 \\ \hline 1 \\ \hline \end{array}$$

Rows sum to one here also!

```
π := | i ← N + M
     | while i ≤ N + M + N − 1
     |     | $\pi_{0,i-(N+M)} \leftarrow \text{Matrix}_{0,i}$
     |     | i ← i + 1
     | π
```

$$\pi = [\ 0.33 \quad 0.551 \quad 0.119\ ]$$

$$\sum_{j=0}^{2} \pi_{0,j} = 1$$

The row sums to one here!

Since all the rows sum to one for a the matrices, we can say that these are valid to work with now.

APPENDIX D: Most Probable Sequences

We need to redefine the FwdProb and scale here because we are working with new matrices. The previous routines worked with the estimates and these will work with the final output matrices.

col := 2

```
FwdProb(col) :=
    i ← 0
    j ← 0
    set ← col
    while i ≤ N − 1
        $\alpha_{i,0} \leftarrow \pi_{0,i} \cdot B_{i,\left(O^{<set>}\right)_0}$
        i ← i + 1
    while j ≤ N − 1
        t ← 0
        while t ≤ T − 1
            $sum \leftarrow \sum_{s=0}^{N-1} \alpha_{s,t} \cdot A_{s,j}$
            $\alpha_{j,t+1} \leftarrow sum \cdot B_{j,\left(O^{<set>}\right)_{t+1}}$
            t ← t + 1
        j ← j + 1
    α
```

```
c(col) :=
    set ← col
    t ← 0
    while t ≤ T
        $c_t \leftarrow \frac{1}{\sum_{i=0}^{N-1} FwdProb(set)_{i,t}}$
        t ← t + 1
    c
```

n := 4 possible observations

r := 3 length of observations

changing the number of observations is easy, but the length will require some work. We would have to add another nested loop to an already hairy code.

The sequence algorithm enumerates all the possible sequences. We are interested here in only three time steps so we will redefine the maximum time value at 2. Remember that MathCad starts its elements at 0 instead of 1.

```
sequence := | q ← 0                                         T := 2
            | i ← 0
            | while i<r
            |    | p_{i,0} ← 0
            |    | i ← i + 1
            | i ← 0
            | j ← 0
            | while j<n^r
            |    | while i<r
            |    |    | seq_{i,j} ← 0
            |    |    | i ← i + 1
            |    | i ← 0
            |    | j ← j + 1
            | i ← 0
            | j ← 0
            | while p_{r-3}<n
            |    | while p_{r-2}<n
            |    |    | while p_{r-1}<n
            |    |    |    | seq^{<q>} ← p
            |    |    |    | p_{r-1} ← p_{r-1} + 1
            |    |    |    | q ← q + 1
            |    |    | p_{r-1} ← 0
            |    |    | p_{r-2} ← p_{r-2} + 1
            |    | p_{r-2} ← 0
            |    | p_{r-3} ← p_{r-3} + 1
            | p_{r-3} ← p_{r-3} - 1
            | q
```

sequence = 64 $n^r = 64$

sequence equals what combinatorics predicts so this routine is correct.

LOGP routine calculates the probability of each sequence. It will be used as sort of a subroutine in the next structure.

```
LOGP(p) := set ← 0
           i ← 0
           j ← 0
           while $i \le N-1$
               $\alpha_{i,0} \leftarrow \pi_{0,i} \cdot B_{i,p_0}$
               $i \leftarrow i+1$
           while $j \le N-1$
               $t \leftarrow 0$
               while $t \le T-1$
                   $sum \leftarrow \sum_{s=0}^{N-1} \alpha_{s,t} \cdot A_{s,j}$
                   $\alpha_{j,t+1} \leftarrow sum \cdot B_{j,p_{t+1}}$
                   $t \leftarrow t+1$
               $j \leftarrow j+1$
           $probab \leftarrow \sum_{i=0}^{N-1} \alpha_{i,T}$
           probab
```

```
Bestset := | q ← 0
           | j ← 0
           | norm ← 0
           | numcol ← $n^r$
           | i ← 0
           | while i<numcol
           |     | $tot_i$ ← 0
           |     | i ← i + 1
           | i ← 0
           | while i<r
           |     | $p_{i,0}$ ← 0
           |     | i ← i + 1
           | while j<numcol
           |     | while i<r
           |     |     | $max_{i,j}$ ← 0
           |     |     | i ← i + 1
           |     | i ← 0
           |     | j ← j + 1
           | i ← 0
           | while $p_{r-3}$<n
           |     | while $p_{r-2}$<n
           |     |     | while $p_{r-1}$<n
           |     |     |     | probability ← LOGP(p)
           |     |     |     | $max^{<q>}$ ← p
           |     |     |     | $tot_q$ ← probability
           |     |     |     | $p_{r-1}$ ← $p_{r-1}$ + 1
           |     |     |     | q ← q + 1
           |     |     | $p_{r-1}$ ← 0
           |     |     | $p_{r-2}$ ← $p_{r-2}$ + 1
           |     | $p_{r-2}$ ← 0
           |     | $p_{r-3}$ ← $p_{r-3}$ + 1
           | $p_{r-3}$ ← $p_{r-3}$ - 1
           | $answer_{0,0}$ ← q
           | normal ← $\sum_{i=0}^{numcol-1} tot_i$
           | $answer_{0,1}$ ← normal
           | i ← 0
           | while i<numcol
```

bestset uses the logP routine to find each sequences probability and then rank order them appropriately

```
while $i<numcol$
    $answer_{1,i} \leftarrow \frac{tot_i}{normal}$
    $i \leftarrow i+1$
$i \leftarrow 2$
$j \leftarrow 0$
while $j<numcol$
    while $i<r+2$
        $answer_{i,j} \leftarrow max_{i-2,j}$
        $i \leftarrow i+1$
    $i \leftarrow 2$
    $j \leftarrow j+1$
answer
```

Here we can output the Bestset routine unsorted.

Bestset =

64	0.78	0	0	0
$7.112\cdot10^{-6}$	$2.669\cdot10^{-4}$	$8.869\cdot10^{-5}$	$7.112\cdot10^{-6}$	$2.676\cdot10^{-4}$
0	0	0	0	0
0	0	0	0	1
0	1	2	3	0

$\text{bestset} := \text{rsort}(\text{Bestset}, 1)$

$\text{bestset} := \text{bestset}^T$

$\text{bestset} := \text{reverse}(\text{bestset})$

$\text{bestset} := \text{bestset}^T$

These statements sort the above matrix according to the probabilities. Then the matrix is output with the highest probability first etc.

This is the stacked Bestset output. Here we can see what movements will most likely occur over the next 3 time iterations.

bestset =

0	0	0	0	0	0	0	0	0
0.377	0.126	0.125	0.124	0.042	0.041	0.041	0.014	0.01
1	2	1	1	2	2	1	2	0
1	1	1	2	1	2	2	2	1
1	1	2	1	2	1	2	2	1

APPENDIX E: Order Calculations

$$\mathrm{realO} = \begin{bmatrix} 1 & 2 & 2 & 2 & 2 & 1 \\ 1 & 0 & 3 & 1 & 1 & 3 \\ 2 & 1 & 2 & 3 & 2 & 2 \end{bmatrix}$$

This data was generated above in Appendix B. It will now be used to "score" the model. We can find the probabilities of these data sequences using the LOGP routine. The routine uses the final output matrices from the training data. If the matrices are perfect then the probabilities should be close to zero on the log scale. Therefore, if we add up all the log probabilities, the lowest score will be the best model. This will be because the matrices were closest to being perfect for that formulation.

$$\mathrm{sightings}_{real} := \log\left(\frac{\mathrm{LOGP}\left(\mathrm{realO}^{\langle real\rangle}\right)}{\mathrm{Bestset}_{0,1}}\right)$$

$$\mathrm{sightings} = \begin{bmatrix} -0.903 \\ -2.476 \\ -2.954 \\ -2.475 \\ -1.379 \\ -2.478 \end{bmatrix}$$

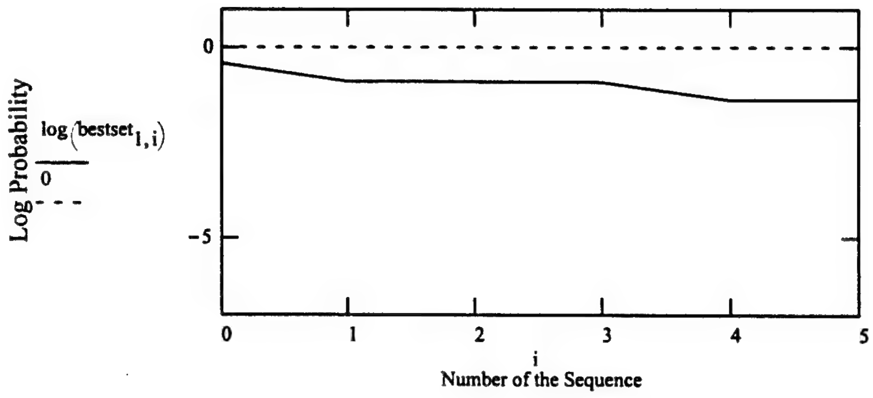
$$i := 0..\mathrm{Real}$$

Here we will sort the sightings from highest probability to smallest probability.

$$\mathrm{sightings} := \mathrm{sort}(\mathrm{sightings})$$

$$\mathrm{sightings} := \mathrm{reverse}(\mathrm{sightings})$$

$$\mathrm{sightings} = \begin{bmatrix} -0.903 \\ -1.379 \\ -2.475 \\ -2.476 \\ -2.478 \\ -2.954 \end{bmatrix}$$



$$\text{Score} := \sum_{i=0}^{\text{Real}} \text{sightings}_i$$

The Rating closest to zero will be the best model.

$\text{Score} = -12.666$

$\text{Real} := 63$

For the 64 possible sequences, the probability of each event summed should add up to one. After normalization it did.

$$\text{prob2} := \sum_{i=0}^{\text{Real}} \text{bestset}_{1,i}$$

$\text{prob2} = 1$

APPENDIX F: Alterations for 6 Observation Symbol Model.

These routines are used for each direction to look at the weighted sum of the highest points over that particular swath. They are unique for each direction.

```
northwest :=
    $x \leftarrow 1$
    while $x \le 3$
        $max_x \leftarrow 0$
        $x \leftarrow x + 1$
    $a \leftarrow 5$
    $b \leftarrow 8$
    $c \leftarrow 1$
    while $a \ge 0$
        $max_c \leftarrow M_{a,b}$ if $|M_{a,b}| > |max_c|$
        $a \leftarrow a - 1$
        $b \leftarrow b + 1$
    $max_c \leftarrow max_c \cdot .5$
    $a \leftarrow 4$
    $b \leftarrow 8$
    $c \leftarrow 2$
    while $a \ge 0$
        $max_c \leftarrow M_{a,b}$ if $|M_{a,b}| > |max_c|$
        $a \leftarrow a - 1$
        $b \leftarrow b + 1$
    $max_c \leftarrow max_c \cdot .25$
    $a \leftarrow 3$
    $b \leftarrow 8$
    $c \leftarrow 3$
    while $a \ge 0$
        $max_c \leftarrow M_{a,b}$ if $|M_{a,b}| > |max_c|$
        $a \leftarrow a - 1$
        $b \leftarrow b + 1$
    $max_c \leftarrow max_c \cdot .25$
    $max \leftarrow \sum_{j=0}^{3} max_j$
    $max \leftarrow .5$ if $max < 0$
```

```
southwest :=
    $x \leftarrow 1$
    while $x \le 3$
        $max_x \leftarrow 0$
        $x \leftarrow x + 1$
    $a \leftarrow 5$
    $b \leftarrow 6$
    $c \leftarrow 1$
    while $a \ge 0$
        $max_c \leftarrow M_{a,b}$ if $|M_{a,b}| > |max_c|$
        $a \leftarrow a - 1$
        $b \leftarrow b - 1$
    $max_c \leftarrow max_c \cdot .5$
    $a \leftarrow 4$
    $b \leftarrow 6$
    $c \leftarrow 2$
    while $a \ge 0$
        $max_c \leftarrow M_{a,b}$ if $|M_{a,b}| > |max_c|$
        $a \leftarrow a - 1$
        $b \leftarrow b - 1$
    $max_c \leftarrow max_c \cdot .25$
    $a \leftarrow 3$
    $b \leftarrow 6$
    $c \leftarrow 3$
    while $a \ge 0$
        $max_c \leftarrow M_{a,b}$ if $|M_{a,b}| > |max_c|$
        $a \leftarrow a - 1$
        $b \leftarrow b - 1$
    $max_c \leftarrow max_c \cdot .25$
    $max \leftarrow \sum_{j=0}^{3} max_j$
    $max \leftarrow .5$ if $max < 0$
```

```
southeast := | x ← 1
             | while x ≤ 3
             |     | $max_x ← 0$
             |     | x ← x + 1
             | a ← 9
             | b ← 6
             | c ← 1
             | while a ≤ 14
             |     | $max_c ← M_{a,b}$  if $|M_{a,b}| > |max_c|$
             |     | a ← a + 1
             |     | b ← b − 1
             | $max_c ← max_c \cdot .5$
             | a ← 10
             | b ← 6
             | c ← 2
             | while a ≤ 14
             |     | $max_c ← M_{a,b}$  if $|M_{a,b}| > |max_c|$
             |     | a ← a + 1
             |     | b ← b − 1
             | $max_c ← max_c \cdot .25$
             | a ← 11
             | b ← 6
             | c ← 3
             | while a ≤ 14
             |     | $max_c ← M_{a,b}$  if $|M_{a,b}| > |max_c|$
             |     | a ← a + 1
             |     | b ← b − 1
             | $max_c ← max_c \cdot .25$
             | $max ← \sum_{j=0}^{3} max_j$
             | max ← .5  if max < 0
```

```
northeast := | x ← 1
             | while x ≤ 3
             |     | $max_x ← 0$
             |     | x ← x + 1
             | a ← 9
             | b ← 8
             | c ← 1
             | while a ≤ 14
             |     | $max_c ← M_{a,b}$  if $|M_{a,b}| > |max_c|$
             |     | a ← a + 1
             |     | b ← b + 1
             | $max_c ← max_c \cdot .5$
             | a ← 10
             | b ← 8
             | c ← 2
             | while a ≤ 14
             |     | $max_c ← M_{a,b}$  if $|M_{a,b}| > |max_c|$
             |     | a ← a + 1
             |     | b ← b + 1
             | $max_c ← max_c \cdot .25$
             | a ← 11
             | b ← 8
             | c ← 3
             | while a ≤ 14
             |     | $max_c ← M_{a,b}$  if $|M_{a,b}| > |max_c|$
             |     | a ← a + 1
             |     | b ← b + 1
             | $max_c ← max_c \cdot .25$
             | $max ← \sum_{j=0}^{3} max_j$
             | max ← .5  if max < 0
```

```
ssouth := | x ← 1
          | while x ≤ 3
          |     | max_x ← 0
          |     | x ← x + 1
          | a ← 7
          | b ← 6
          | c ← 1
          | while b ≥ 0
          |     | max_c ← M_{a,b}  if |M_{a,b}| > |max_c|
          |     | b ← b − 1
          | max_c ← max_c · .5
          | a ← 6
          | b ← 6
          | c ← 2
          | while b ≥ 0
          |     | max_c ← M_{a,b}  if |M_{a,b}| > |max_c|
          |     | b ← b − 1
          | max_c ← max_c · .25
          | a ← 8
          | b ← 6
          | c ← 3
          | while b ≥ 0
          |     | max_c ← M_{a,b}  if |M_{a,b}| > |max_c|
          |     | b ← b − 1
          | max_c ← max_c · .25
          | max ← $\sum_{j=0}^{3} max_j$
          | max ← .5  if max < 0
```

```
nnorth := | x ← 1
          | while x ≤ 3
          |     | max_x ← 0
          |     | x ← x + 1
          | a ← 7
          | b ← 8
          | c ← 1
          | while b ≤ 14
          |     | max_c ← M_{a,b}  if |M_{a,b}| > |max_c|
          |     | b ← b + 1
          | max_c ← max_c · .5
          | a ← 6
          | b ← 8
          | c ← 2
          | while b ≤ 14
          |     | max_c ← M_{a,b}  if |M_{a,b}| > |max_c|
          |     | b ← b + 1
          | max_c ← max_c · .25
          | a ← 8
          | b ← 8
          | c ← 3
          | while b ≤ 14
          |     | max_c ← M_{a,b}  if |M_{a,b}| > |max_c|
          |     | b ← b + 1
          | max_c ← max_c · .25
          | max ← $\sum_{j=0}^{3} max_j$
          | max ← .5  if max < 0
```

northwest = 0.5

ssouth = 3.792

southeast = 5.102

southwest = 2.554

nnorth = 8.339

northeast = 3.62

$\text{totconceal} := \text{nnorth} + \text{northeast} + \text{southeast} + \text{ssouth} + \text{southwest} + \text{northwest}$

$\text{totconceal} = 23.908$

totconceal will be used as the normalizer once all the respective scores are added up.

for conceal state:

$$\frac{\text{nnorth}}{\text{totconceal}} = 0.349$$

$$\frac{\text{northwest}}{\text{totconceal}} = 0.021 \qquad \frac{\text{northeast}}{\text{totconceal}} = 0.151$$

^

$$\frac{\text{southwest}}{\text{totconceal}} = 0.107 \qquad \frac{\text{southeast}}{\text{totconceal}} = 0.213$$

$$\frac{\text{ssouth}}{\text{totconceal}} = 0.159$$

for speed state:

$\text{totspd} := \text{northwest}^{-1} + \text{northeast}^{-1} + \text{southeast}^{-1} + \text{nnorth}^{-1} + \text{ssouth}^{-1} + \text{southwest}^{-1}$

$\text{totspd} = 3.247$

$$\frac{\text{nnorth}^{-1}}{\text{totspd}} = 0.037$$

$$\frac{\text{northwest}^{-1}}{\text{totspd}} = 0.616 \qquad \frac{\text{northeast}^{-1}}{\text{totspd}} = 0.085$$

^

$$\frac{\text{southwest}^{-1}}{\text{totspd}} = 0.121 \qquad \frac{\text{southeast}^{-1}}{\text{totspd}} = 0.06$$

$$\frac{\text{ssouth}^{-1}}{\text{totspd}} = 0.081$$

let 0 be the conceal state and 2 be the speed state. NNorth:0 northeast:1 southeast:2 SSouth:3 southwest:4 northwest:5. Keep in mind here that the number of states doesn't change from 3. That is why there are 3 rows in the B matrix still.

$$B := \begin{bmatrix} \frac{nnorth}{totconceal} & \frac{northeast}{totconceal} & \frac{southeast}{totconceal} & \frac{ssouth}{totconceal} & \frac{southwest}{totconceal} & \frac{northwest}{totconceal} \\ .166667 & .166667 & .166667 & .166667 & .166667 & .166667 \\ \frac{nnorth^{-1}}{totspd} & \frac{northeast^{-1}}{totspd} & \frac{southeast^{-1}}{totspd} & \frac{ssouth^{-1}}{totspd} & \frac{southwest^{-1}}{totspd} & \frac{northwest^{-1}}{totspd} \end{bmatrix}$$

$$B = \begin{bmatrix} 0.349 & 0.151 & 0.213 & 0.159 & 0.107 & 0.021 \\ 0.167 & 0.167 & 0.167 & 0.167 & 0.167 & 0.167 \\ 0.037 & 0.085 & 0.06 & 0.081 & 0.121 & 0.616 \end{bmatrix}$$

This will be the initial B matrix we use to train the model. Notice the uniform distribution in the ambiguous state.

$w := 0 .. 2$ Define a range variable to sum rows of the B matrix.

$$\sum_{j=0}^{5} B_{w,j}$$

All the rows still sum to one.

$$\boxed{\begin{array}{|c|}\hline 1 \\ \hline 1 \\ \hline 1 \\ \hline\end{array}}$$

$M := 6$ define the number of observations possible.

From here on down Appendix A will pick up nicely since nothing else changes.

APPENDIX G: Alterations for 4 Markov State Model.

The number of symbols reverts back to 4 here for symplicity.

$$f(x,\alpha,\beta) := \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\cdot\Gamma(\beta)}\cdot x^{\alpha-1}\cdot(1-x)^{\beta-1}$$

The beta distribution is here because that is how we plan to estimate the changed states.

$N := 4$

now that we have more than 3 states, defining will be allittle more difficult. so, let's just use the beta to fill in the blanks.

$$\xi(v,\alpha,\beta) := \int_{(v-1)\cdot\frac{1}{N}}^{v\cdot\left(\frac{1}{N}\right)} f(x,\alpha,\beta)\,dx$$

let 0 be the conceal state and 2 be the speed state. North:0 east:1 south:2 west:3

$$B := \begin{bmatrix} \frac{north}{totconceal} & \frac{east}{totconceal} & \frac{south}{totconceal} & \frac{west}{totconceal} \\ \xi(1,2,4) & \xi(2,2,4) & \xi(3,2,4) & \xi(4,2,4) \\ \xi(1,4,2) & \xi(2,4,2) & \xi(3,4,2) & \xi(4,4,2) \\ \frac{north^{-1}}{totspd} & \frac{east^{-1}}{totspd} & \frac{south^{-1}}{totspd} & \frac{west^{-1}}{totspd} \end{bmatrix}$$

We have replaced the ambivalent state with 2 semi-ambivalent states. There is really no physical significance that we can determine now but that doesn't matter. The parameters are set up to compliment the other distribution..

$$B = \begin{bmatrix} 0.44 & 0.296 & 0.216 & 0.048 \\ 0.367 & 0.445 & 0.172 & 0.016 \\ 0.016 & 0.172 & 0.445 & 0.367 \\ 0.073 & 0.109 & 0.148 & 0.67 \end{bmatrix}$$

$w := 0..N-1$

$$\sum_{j=0}^{3} B_{w,j}$$

$$\begin{array}{|c|}\hline 1 \\ \hline 1 \\ \hline 1 \\ \hline 1 \\ \hline\end{array}$$

All rows sum to one.

This will be the initial B matrix we use to train the model. Notice the uniform distribution in the ambiguous state.

$M := 4$ define the number of observations possible.

$N := 4$

$i := 0 .. N-1$

$j := 0 .. N-1$

$\alpha := 3$

$\beta := 4$

$$f(x,\alpha,\beta) := \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\cdot\Gamma(\beta)}\cdot x^{\alpha-1}\cdot(1-x)^{\beta-1}$$

Now we have to find the state transition matrix first guess. Beta distribution will be used because it normalizes to one and is easily parameterized. We only need three integrations because there are only three states to transition to.

$$\int_{0}^{\frac{1}{N}} f(x,\alpha,\beta)\,dx = 0.169$$

$$\int_{\frac{1}{N}}^{2\cdot\left(\frac{1}{N}\right)} f(x,\alpha,\beta)\,dx = 0.487$$

$$\int_{2\cdot\left(\frac{1}{N}\right)}^{3\cdot\left(\frac{1}{N}\right)} f(x,\alpha,\beta)\,dx = 0.306$$

$$\int_{3\cdot\left(\frac{1}{N}\right)}^{4\cdot\left(\frac{1}{N}\right)} f(x,\alpha,\beta)\,dx = 0.038$$

When doing sample calculations now an extra integartion is needed because we have an extra transition possible.

$$\xi(v,\alpha,\beta) := \int_{(v-1)\cdot\frac{1}{N}}^{v\cdot\left(\frac{1}{N}\right)} f(x,\alpha,\beta)\,dx \qquad \xi(3,\alpha,\beta) = 0.306$$

$$A := \begin{bmatrix} \xi(1,2,4) & \xi(2,2,4) & \xi(3,2,4) & \xi(4,2,4) \\ \xi(1,4,3) & \xi(2,4,3) & \xi(3,4,3) & \xi(4,4,3) \\ \xi(1,3,4) & \xi(2,3,4) & \xi(3,3,4) & \xi(4,3,4) \\ \xi(1,4,2) & \xi(2,4,2) & \xi(3,4,2) & \xi(4,4,2) \end{bmatrix} \quad A = \begin{bmatrix} 0.367 & 0.445 & 0.172 & 0.016 \\ 0.038 & 0.306 & 0.487 & 0.169 \\ 0.169 & 0.487 & 0.306 & 0.038 \\ 0.016 & 0.172 & 0.445 & 0.367 \end{bmatrix}$$

$$A(a,b,c,d,e,f,g,h,N) := \begin{bmatrix} \xi(N-3,a,b) & \xi(N-2,a,b) & \xi(N-1,a,b) & \xi(N,a,b) \\ \xi(N-3,c,d) & \xi(N-2,c,d) & \xi(N-1,c,d) & \xi(N,c,d) \\ \xi(N-3,e,f) & \xi(N-2,e,f) & \xi(N-1,e,f) & \xi(N,e,f) \\ \xi(N-3,g,h) & \xi(N-2,g,h) & \xi(N-1,g,h) & \xi(N,g,h) \end{bmatrix}$$

$$A(2,4,4,3,3,4,4,2,4) = \begin{bmatrix} 0.367 & 0.445 & 0.172 & 0.016 \\ 0.038 & 0.306 & 0.487 & 0.169 \\ 0.169 & 0.487 & 0.306 & 0.038 \\ 0.016 & 0.172 & 0.445 & 0.367 \end{bmatrix}$$

$A := A(2,4,4,3,3,4,4,2,4)$

All the above manipulation was either putting it into an easy form MathCad wil agree with or checking things out with myself that they actually work.

$$A = \begin{bmatrix} 0.367 & 0.445 & 0.172 & 0.016 \\ 0.038 & 0.306 & 0.487 & 0.169 \\ 0.169 & 0.487 & 0.306 & 0.038 \\ 0.016 & 0.172 & 0.445 & 0.367 \end{bmatrix}$$

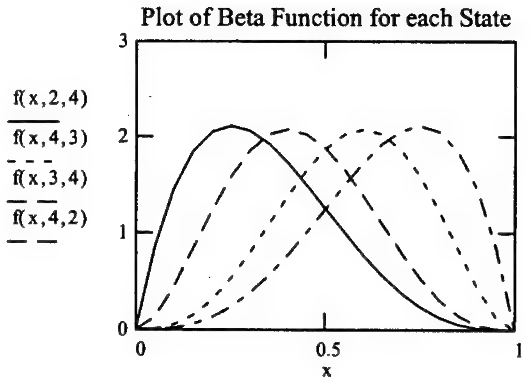
notice we set up the initial matrix so the chance of reentering the same state is the highest.

$$\sum_{j=0}^{3} A_{w,j}$$

$$\begin{array}{|c|} \hline 1 \\ \hline 1 \\ \hline 1 \\ \hline 1 \\ \hline \end{array}$$

All rows still sum to one so we know this is a valid matrix.

$x := 0, .05 .. 1$



here we have the initial beta functions with parameters for the state transition matrix. Now there is simply an extra state to deal with.

Now the initial state probability matrix has to be determined. It is the same as the state transition matrix but, it only has one row.

The procedure is the same here as in Appendix A, basically. The difference is the extra element to account for the extra Markov state.

$\alpha := 2$

$\beta := 2$

$$f(x,\alpha,\beta) := \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\cdot\Gamma(\beta)}\cdot x^{\alpha-1}\cdot(1-x)^{\beta-1}$$

$$\int_{0}^{\frac{1}{N}} f(x,\alpha,\beta)\,dx = 0.156 \qquad \int_{\frac{1}{N}}^{2\cdot\left(\frac{1}{N}\right)} f(x,\alpha,\beta)\,dx = 0.344$$

$$\int_{2\cdot\left(\frac{1}{N}\right)}^{3\cdot\left(\frac{1}{N}\right)} f(x,\alpha,\beta)\,dx = 0.344 \qquad \int_{3\cdot\left(\frac{1}{N}\right)}^{4\cdot\left(\frac{1}{N}\right)} f(x,\alpha,\beta)\,dx = 0.156$$

$$\xi(v,\alpha,\beta) := \int_{(v-1)\cdot\frac{1}{N}}^{v\cdot\left(\frac{1}{N}\right)} f(x,\alpha,\beta)\,dx \qquad \xi(3,\alpha,\beta) = 0.344$$

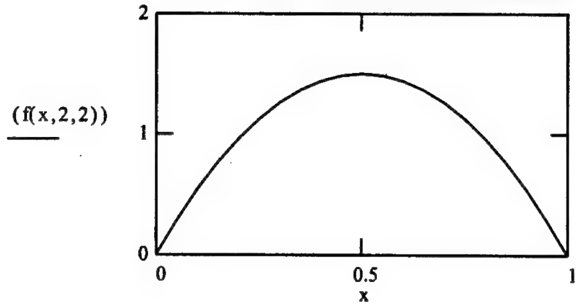
$\pi := (\xi(1,2,2)\ \ \xi(2,2,2)\ \ \xi(3,2,2)\ \ \xi(4,2,2))$ $\quad \pi = [\,0.156\ \ 0.344\ \ 0.344\ \ 0.156\,]$

$\pi(a,b,N) := (\xi(N-3,a,b)\ \ \xi(N-2,a,b)\ \ \xi(N-1,a,b)\ \ \xi(N,a,b))$

$\pi(2,2,4) = [\,0.156\ \ 0.344\ \ 0.344\ \ 0.156\,]$

$x := 0, .05 .. 1$ $\qquad \pi := \pi(2,2,4)$

Plot of Beta Function for Initial State



here we have the beta function with parameter for the initial state probability matrix.

$$\sum_{j=0}^{3} \pi_{0,j} = 1$$

APPENDIX H: Alterations for 5 Markov State Model.

$$f(x,\alpha,\beta) := \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\cdot\Gamma(\beta)}\cdot x^{\alpha-1}\cdot(1-x)^{\beta-1}$$

The number of symbols is going to be 4 here also for simplicity sake.

$N := 4$ here N is just the number of integrations we have to do. For right now, it doesn't stand for the number of Markov states possible.

now that we have more than 3 states, defining will be allittle more difficult. so, let's just use the handy dandy beta to fill in the blanks.

$$\xi(v,\alpha,\beta) := \int_{(v-1)\cdot\frac{1}{N}}^{v\cdot\left(\frac{1}{N}\right)} f(x,\alpha,\beta)\,dx$$

let 0 be the conceal state and 2 be the speed state. North:0 east:1 south:2 west:3

$$B := \begin{bmatrix} \frac{north}{totconceal} & \frac{east}{totconceal} & \frac{south}{totconceal} & \frac{west}{totconceal} \\ \xi(1,2,4) & \xi(2,2,4) & \xi(3,2,4) & \xi(4,2,4) \\ .25 & .25 & .25 & .25 \\ \xi(1,4,2) & \xi(2,4,2) & \xi(3,4,2) & \xi(4,4,2) \\ \frac{north^{-1}}{totspd} & \frac{east^{-1}}{totspd} & \frac{south^{-1}}{totspd} & \frac{west^{-1}}{totspd} \end{bmatrix}$$

Here we added a true ambivalent state back into the model. How we choose to estimate these states is a creative choice and this is what we have choosen.

$w := 0..N$

$$B = \begin{bmatrix} 0.44 & 0.296 & 0.216 & 0.048 \\ 0.367 & 0.445 & 0.172 & 0.016 \\ 0.25 & 0.25 & 0.25 & 0.25 \\ 0.016 & 0.172 & 0.445 & 0.367 \\ 0.073 & 0.109 & 0.148 & 0.67 \end{bmatrix} \qquad \sum_{j=0}^{3} B_{w,j} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

All rows still sum to one.

This will be the initial B matrix we use to train the model. Notice the uniform distribution in the ambivalent state.

$M := 4$ define the number of observations possible.

Now we let N be the states allowed in the model.

$N := 5$

$\alpha := 3$

$i := 0..N-1$

$\beta := 4$

$j := 0..N-1$

$f(x,\alpha,\beta) := \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\cdot\Gamma(\beta)}\cdot x^{\alpha-1}\cdot(1-x)^{\beta-1}$

Now we have to find the state transition matrix first guess. Beta distribution will be used because it normalizes to one and is easily parameterized. We only need three integrations because there are only three states to transition to.

$$\int_0^{\frac{1}{N}} f(x,\alpha,\beta)\,dx = 0.099$$

$$\int_{\frac{1}{N}}^{2\cdot\left(\frac{1}{N}\right)} f(x,\alpha,\beta)\,dx = 0.357$$

$$\int_{2\cdot\left(\frac{1}{N}\right)}^{3\cdot\left(\frac{1}{N}\right)} f(x,\alpha,\beta)\,dx = 0.365$$

$$\int_{3\cdot\left(\frac{1}{N}\right)}^{4\cdot\left(\frac{1}{N}\right)} f(x,\alpha,\beta)\,dx = 0.162$$

$$\int_{4\cdot\left(\frac{1}{N}\right)}^{5\cdot\left(\frac{1}{N}\right)} f(x,\alpha,\beta)\,dx = 0.017$$

Notice again that the number of sample calculations has to increase by one because of the extra state.

$$\xi(v,\alpha,\beta) := \int_{(v-1)\cdot\frac{1}{N}}^{v\cdot\left(\frac{1}{N}\right)} f(x,\alpha,\beta)\,dx \qquad \xi(3,\alpha,\beta) = 0.365$$

$$A := \begin{bmatrix} \xi(1,2,4) & \xi(2,2,4) & \xi(3,2,4) & \xi(4,2,4) & \xi(5,2,4) \\ \xi(1,4,3) & \xi(2,4,3) & \xi(3,4,3) & \xi(4,4,3) & \xi(5,4,3) \\ \xi(1,3.5,3.5) & \xi(2,3.5,3.5) & \xi(3,3.5,3.5) & \xi(4,3.5,3.5) & \xi(5,3.5,3.5) \\ \xi(1,3,4) & \xi(2,3,4) & \xi(3,3,4) & \xi(4,3,4) & \xi(5,3,4) \\ \xi(1,4,2) & \xi(2,4,2) & \xi(3,4,2) & \xi(4,4,2) & \xi(5,4,2) \end{bmatrix}$$

$$A = \begin{bmatrix} 0.263 & 0.4 & 0.25 & 0.08 & 6.72\cdot 10^{-3} \\ 0.017 & 0.162 & 0.365 & 0.357 & 0.099 \\ 0.044 & 0.259 & 0.394 & 0.259 & 0.044 \\ 0.099 & 0.357 & 0.365 & 0.162 & 0.017 \\ 6.72\cdot 10^{-3} & 0.08 & 0.25 & 0.4 & 0.263 \end{bmatrix}$$

$$A(a,b,c,d,e,f,g,h,N) := \begin{bmatrix} \xi(N-3,a,b) & \xi(N-2,a,b) & \xi(N-1,a,b) & \xi(N,a,b) \\ \xi(N-3,c,d) & \xi(N-2,c,d) & \xi(N-1,c,d) & \xi(N,c,d) \\ \xi(N-3,e,f) & \xi(N-2,e,f) & \xi(N-1,e,f) & \xi(N,e,f) \\ \xi(N-3,g,h) & \xi(N-2,g,h) & \xi(N-1,g,h) & \xi(N,g,h) \end{bmatrix}$$

$$A(a,b,c,d,e,f,g,h,i,j,N) := \begin{bmatrix} \xi(N-4,a,b) & \xi(N-3,a,b) & \xi(N-2,a,b) & \xi(N-1,a,b) & \xi(N,a,b) \\ \xi(N-4,c,d) & \xi(N-3,c,d) & \xi(N-2,c,d) & \xi(N-1,c,d) & \xi(N,c,d) \\ \xi(N-4,e,f) & \xi(N-3,e,f) & \xi(N-2,e,f) & \xi(N-1,e,f) & \xi(N,e,f) \\ \xi(N-4,g,h) & \xi(N-3,g,h) & \xi(N-2,g,h) & \xi(N-1,g,h) & \xi(N,g,h) \\ \xi(N-4,i,j) & \xi(N-3,i,j) & \xi(N-2,i,j) & \xi(N-1,i,j) & \xi(N,i,j) \end{bmatrix}$$

$$A(2,4,4,3,3.5,3.5,3,4,4,2,5) = \begin{bmatrix} 0.263 & 0.4 & 0.25 & 0.08 & 6.72\cdot 10^{-3} \\ 0.017 & 0.162 & 0.365 & 0.357 & 0.099 \\ 0.044 & 0.259 & 0.394 & 0.259 & 0.044 \\ 0.099 & 0.357 & 0.365 & 0.162 & 0.017 \\ 6.72\cdot 10^{-3} & 0.08 & 0.25 & 0.4 & 0.263 \end{bmatrix}$$

$A := A(2,4,4,3,3.5,3.5,3,4,4,2,5)$ All the above manipulation was either putting it into an easy form MathCad wil agree with or checking things out with myself that they actually work.

$$A = \begin{bmatrix} 0.263 & 0.4 & 0.25 & 0.08 & 6.72\cdot 10^{-3} \\ 0.017 & 0.162 & 0.365 & 0.357 & 0.099 \\ 0.044 & 0.259 & 0.394 & 0.259 & 0.044 \\ 0.099 & 0.357 & 0.365 & 0.162 & 0.017 \\ 6.72\cdot 10^{-3} & 0.08 & 0.25 & 0.4 & 0.263 \end{bmatrix}$$

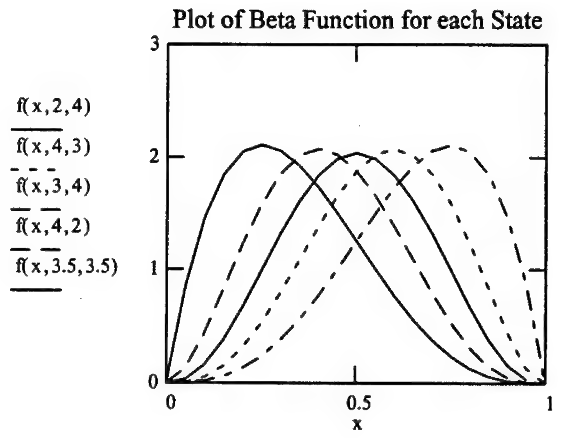
notice we set up the initial matrix so the chance of reentering the same state is the highest.

$$\sum_{j=0}^{4} A_{w,j}$$

$$\begin{array}{|c|}\hline 1\\\hline 1\\\hline 1\\\hline 1\\\hline 1\\\hline\end{array}$$

All rows still sum to one indicating that the matrix is valid.

$x := 0, .05 .. 1$



here we have the initial beta functions with parameters for the state transition matrix.

Now the initial state probability matrix has to be determined. It is the same as the state transition matrix but, it only has one row.

The procedure is the same here as in Appendix A, basically. The difference is the extra two elements to account for the extra Markov states.

$\alpha := 2$

$\beta := 2$

$$f(x,\alpha,\beta) := \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\cdot\Gamma(\beta)}\cdot x^{\alpha-1}\cdot(1-x)^{\beta-1}$$

$$\int_{0}^{\frac{1}{N}} f(x,\alpha,\beta)\,dx = 0.104 \qquad \int_{\frac{1}{N}}^{2\cdot\left(\frac{1}{N}\right)} f(x,\alpha,\beta)\,dx = 0.248$$

$$\int_{2\cdot\left(\frac{1}{N}\right)}^{3\cdot\left(\frac{1}{N}\right)} f(x,\alpha,\beta)\,dx = 0.296 \qquad \int_{3\cdot\left(\frac{1}{N}\right)}^{4\cdot\left(\frac{1}{N}\right)} f(x,\alpha,\beta)\,dx = 0.248$$

$$\int_{4\cdot\left(\frac{1}{N}\right)}^{5\cdot\left(\frac{1}{N}\right)} f(x,\alpha,\beta)\,dx = 0.104$$

$$\xi(v,\alpha,\beta) := \int_{(v-1)\cdot\frac{1}{N}}^{v\cdot\left(\frac{1}{N}\right)} f(x,\alpha,\beta)\,dx \qquad \xi(3,\alpha,\beta) = 0.296$$

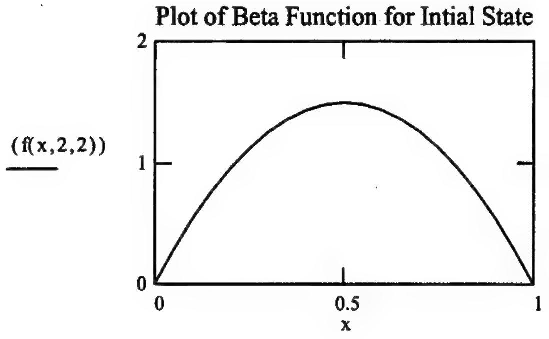
$\pi := (\xi(1,2,2)\ \ \xi(2,2,2)\ \ \xi(3,2,2)\ \ \xi(4,2,2)\ \ \xi(5,2,2))$

$\pi = [\ 0.104\ \ \ 0.248\ \ \ 0.296\ \ \ 0.248\ \ \ 0.104\ ]$

$\pi(a,b,N) := (\xi(N-4,a,b)\ \ \xi(N-3,a,b)\ \ \xi(N-2,a,b)\ \ \xi(N-1,a,b)\ \ \xi(N,a,b))$

$\pi(2,2,5) = [\, 0.104 \quad 0.248 \quad 0.296 \quad 0.248 \quad 0.104 \,]$

$x := 0, .05 .. 1$ $\qquad$ $\pi := \pi(2,2,5)$



here we have the beta function with parameter for the initial state probability matrix.

$$\sum_{j=0}^{4} \pi_{0,j} = 1$$

All rows still sum to one.

APPENDIX I: Correctness of Model.

Matrices for trivial sequence 0,0,0

$$A = \begin{bmatrix} 0.72 & 0.273 & 7.346\bullet 10^{-3} \\ 0.409 & 0.544 & 0.047 \\ 0.153 & 0.653 & 0.193 \end{bmatrix} \qquad B = \begin{bmatrix} 0.943 & 0.019 & 0.019 & 0.019 \\ 0.943 & 0.019 & 0.019 & 0.019 \\ 0.943 & 0.019 & 0.019 & 0.019 \end{bmatrix}$$

$$\pi = \begin{bmatrix} 0.572 & 0.394 & 0.034 \end{bmatrix}$$

It should be obvious that the matrices trained directly to the conceal state(0). The highest probabilities are in the recurrent conceal state and transitions to the conceal state for the A matrix. The B matrix obviously favors north. π favors the conceal state initially.

$$\gamma = \begin{bmatrix} 0.572 & 0.613 & 0.497 \\ 0.394 & 0.365 & 0.464 \\ 0.034 & 0.022 & 0.039 \end{bmatrix}$$

The expected states all fall to the conceal state.

bestset =

64	0.816	0	0	0	0	0	0	0
0.84	0.017	0.017	0.017	0.017	0.017	0.017	0.017	0.017
0	0	0	2	1	0	0	0	3
0	0	0	0	0	2	1	0	0
0	1	3	0	0	0	0	2	0

The highest probabilities favor north and the conceal state. This is obviously so for the first column.

Matrices for trivial sequence 3,3,0

$$A = \begin{bmatrix} 0.488 & 0.454 & 0.059 \\ 0.226 & 0.577 & 0.197 \\ 0.062 & 0.46 & 0.478 \end{bmatrix} \qquad B = \begin{bmatrix} 0.463 & 0.019 & 0.019 & 0.498 \\ 0.421 & 0.019 & 0.019 & 0.541 \\ 0.23 & 0.019 & 0.019 & 0.731 \end{bmatrix}$$

$$\pi = \begin{bmatrix} 0.022 & 0.33 & 0.649 \end{bmatrix}$$

It should be obvious that the matrices trained directly to the speed state(0). The highest probabilities are in the recurrent speed state for the A matrix. The B matrix obviously favors the observed data. π favors the speed state initially.

$$\gamma = \begin{bmatrix} 0.022 & 0.029 & 0.047 \\ 0.33 & 0.404 & 0.571 \\ 0.649 & 0.567 & 0.383 \end{bmatrix}$$

The expected states all fall to the speed state for the first two iterations. After that, some indecision is introduced and then the state is expected to go ambivalent. The results make sense given the input data.

bestset =

0	0	0	0	0	0	0	64
0.242	0.148	0.127	0.109	0.088	0.07	0.06	0.044
3	3	3	0	3	0	0	0
3	0	3	3	0	0	3	0
3	3	0	3	0	3	0	0

The highest probabilities favor west and the speed state because west represents lowest areas of the field. This is obviously so for the first column probability.

Matrices for trivial sequence 0,1,2

$$A = \begin{bmatrix} 0.561 & 0.417 & 0.022 \\ 0.239 & 0.645 & 0.116 \\ 0.064 & 0.572 & 0.363 \end{bmatrix} \qquad B = \begin{bmatrix} 0.414 & 0.355 & 0.212 & 0.02 \\ 0.279 & 0.317 & 0.385 & 0.02 \\ 0.209 & 0.253 & 0.518 & 0.02 \end{bmatrix}$$

$$\pi = \begin{bmatrix} 0.502 & 0.446 & 0.052 \end{bmatrix}$$

It should be obvious that the matrices trained to no specific state. The highest probabilities are in the recurrent states for the A matrix. The B matrix obviously favors the observed data. π doesn't favor any particular state either.

$$\gamma = \begin{bmatrix} 0.502 & 0.43 & 0.257 \\ 0.446 & 0.507 & 0.615 \\ 0.052 & 0.063 & 0.128 \end{bmatrix}$$

The expected states favor conceal and ambivalent because the circle it made headed towards the conceal initially. We can't really pin down a state here because there was no definite pattern. The object would stay in the ambivalent state.

bestset =

0	0	0	0.822	0	0	0	64	0	0
1	0.041	0.04	0.039	0.039	0.039	0.039	0.039	0.038	0.038
2	0	0	0	1	1	0	0	1	0
3	0	1	0	0	1	2	0	2	1
4	2	2	1	2	2	2	0	2	1

There are really no high probabilities here. The object could move one of several ways. This is more evidence that the object is ambivalent as to were he wants to go.

APPENDIX J: MODEL ORDER

First we will look at the models with 4 observation symbols allowed.

$$\text{realO} := \begin{pmatrix} 1 & 2 & 2 & 2 & 2 & 1 \\ 1 & 0 & 3 & 1 & 1 & 3 \\ 2 & 1 & 2 & 3 & 2 & 2 \end{pmatrix}$$

This is the data that was used to test each model. The LOGP routine found the probability of each sequence from left to right across the realO matrix. Each probability was converted to log form. These log probabilities are the output of the model.

$\text{Real} := 5$

$\text{real} := 0..\text{Real}$

The S refers to the number of states, while the O refers to the number of observation symbols for the model.

$$\text{State3\_Obs4} := \begin{bmatrix} -1.174 \\ -1.297 \\ -1.409 \\ -1.472 \\ -1.556 \\ -2.647 \end{bmatrix} \qquad \text{State4\_Obs4} := \begin{bmatrix} -.806 \\ -.997 \\ -1.438 \\ -1.585 \\ -1.851 \\ -2.67 \end{bmatrix}$$

$$\text{Score\_S3\_O4} := \sum_{i=0}^{\text{Real}} \text{State3\_Obs4}_i \qquad \text{Score\_S4\_O4} := \sum_{i=0}^{\text{Real}} \text{State4\_Obs4}_i$$

$\text{Score\_S3\_O4} = -9.555$ $\qquad$ $\text{Score\_S4\_O4} = -9.347$

$$\text{State5\_Obs4} := \begin{bmatrix} -.873 \\ -1.085 \\ -1.379 \\ -1.413 \\ -1.726 \\ -2.693 \end{bmatrix}$$

As we mentioned before, the lowest score will show the most appropriate model given the realO data.

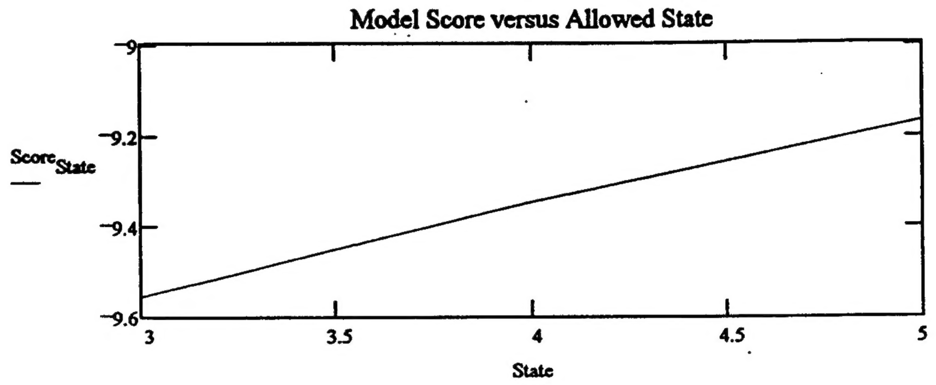
$$\text{Score\_S5\_O4} := \sum_{i=0}^{\text{Real}} \text{State5\_Obs4}_i$$

$\text{Score\_S5\_O4} = -9.169$

$\text{Score}_3 := \text{Score\_S3\_O4}$

$\text{Score}_4 := \text{Score\_S4\_O4}$

$\text{Score}_5 := \text{Score\_S5\_O4}$ $\text{State} := 3 .. 5$



It is obvious that as the state increases the model becomes more appropriate. Here the 5 state model would be best because it has the lowest score. In other words, the 5 state model matrices are closest to being ideal so we should use that model. The order for the 4 observational symbol models is 5.

Next we will look at the models with 6 observation symbols allowed.

$$\text{realO} := \begin{pmatrix} 1 & 2 & 2 & 1 & 2 & 1 \\ 1 & 0 & 1 & 1 & 2 & 3 \\ 2 & 1 & 1 & 3 & 2 & 2 \end{pmatrix}$$

This is the data that was used to test each model. The LOGP routine found the probability of each sequence from left to right across the realO matrix. Each probability was converted to log form. These log probabilities are the output of the model.

Real := 5

real := 0.. Real

The S refers to the number of states, while the O refers to the number of observation symbols for the model.

$$\text{State3\_Obs6} := \begin{bmatrix} -1.847 \\ -1.851 \\ -1.858 \\ -1.874 \\ -1.874 \\ -1.881 \end{bmatrix} \qquad \text{State4\_Obs6} := \begin{bmatrix} -1.583 \\ -1.589 \\ -1.734 \\ -1.93 \\ -1.997 \\ -2.032 \end{bmatrix}$$

$$\text{Score\_S3\_O6} := \sum_{i=0}^{\text{Real}} \text{State3\_Obs6}_i \qquad \text{Score\_S4\_O6} := \sum_{i=0}^{\text{Real}} \text{State4\_Obs6}_i$$

Score_S3_O6 = −11.185 Score_S4_O6 = −10.865

$$\text{State5\_Obs6} := \begin{bmatrix} -1.527 \\ -1.529 \\ -1.714 \\ -1.903 \\ -2.04 \\ -2.086 \end{bmatrix}$$

As we mentioned before, the lowest score will show the most appropriate model given the realO data.

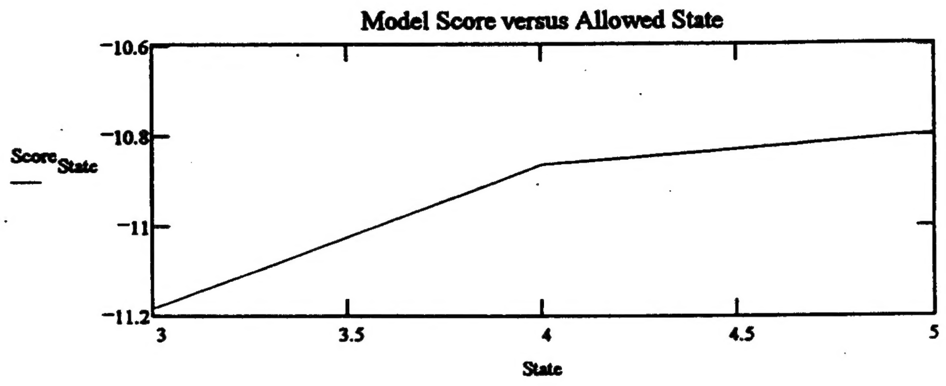
$$\text{Score\_S5\_O6} := \sum_{i=0}^{\text{Real}} \text{State5\_Obs6}_i$$

Score_S5_O6 = −10.799

$\text{Score}_3 := \text{Score\_S3\_O6}$

$\text{Score}_4 := \text{Score\_S4\_O6}$

$\text{Score}_5 := \text{Score\_S5\_O6}$ $\text{State} := 3 .. 5$



It is obvious that as the state increases the model becomes more appropriate. Although there is a difference from the 6 symbol versus the 4 symbol model. The 4 state model here would probably serve just as well as the 5 state model because there is such a small change. Strictly speaking, however, the 5 state model would still be best.